08-30-00 A

Customer No. 20350
TOWNSEND and TOWNSEND and CREW LLP
Two Embarcadero Center, 8th Floor
San Francisco, California 94111-3834
(415) 576-0200

ASSISTANT COMMISSIONER FOR PATENTS
BOX PATENT APPLICATION
Washington, D.C. 20231

Attorney Docket No. 17887-005320US

"Express Mail" Label No. EL623997245US

Date of Deposit: August 29, 2000

Sir:
Transmitted herewith for filing under 37 CFR 1.53(b) is the

[X] patent application of

Inventor(s)/Applicant Identifier: Greg I. Chiou et al.

For: MOBILE SOFTWARE MORPHING AGENT

[X] This application claims priority from each of the following Application Nos./filing dates:
60/212,060/ filed June 16, 2000,
the disclosure of which is incorporated herein by reference.

Enclosed are:

[X] 9 page(s) of specification
[X] 5 page(s) of claims
[X] 1 page of Abstract
[X] 3 sheet(s) of [ ] formal [X] informal drawing(s).
[X] A [ ] signed [X] unsunsigned Declaration.
[X] APPENDIX A (26 pages)
[X] APPENDIX B1 (87 pages)
[X] APPENDIX B2 (14 pages)
[X] APPENDIX B3 (4 pages)
[X] APPENDIX C (3 pages)

	(Col. 1)		(Col. 2)
FOR:	NO. FILED		NO. EXTRA
BASIC FEE			
TOTAL CLAIMS	23	- 20 =	*3
INDEP. CLAIMS	2	- 3 =	*0
[ ] MULTIPLE DEPENDENT CLAIM PRESENTED			

-* If the difference in Col. 1 is less than 0, enter "0" in Col. 2.

SMALL ENTITY			OTHER THAN SMALL ENTITY	
RATE	FEE	OR	RATE	FEE
	$345.00	OR		$690.00
x $9.00 =		OR	x $18.00 =	$54.00
x $39.00 =		OR	x $78.00 =	$0.00
+ $130.00 =		OR	+ $260.00 =	
TOTAL		OR	TOTAL	$744.00

Please charge Deposit Account No. 20-1430 as follows:
[X] Filing fee $ $744.00
[X] Any additional fees associated with this paper or during the pendency of this application.
[ ] The issue fee set in 37 CFR 1.18 at or before mailing of the Notice of Allowance, pursuant to 37 CFR 1.311(b)

[ ] A check for $_____ is enclosed.
2 extra copies of this sheet are enclosed.

Respectfully submitted,
TOWNSEND and TOWNSEND and CREW LLP

Gerald T. Gray
Reg No.: 41,797
Attorneys for Applicant

Telephone:
(415) 576-0200

Facsimile:
(415) 576-0300

SF 1130522 v1

Attorney Docket No.: 17887-005320US

PATENT APPLICATION

Mobile Software Morphing Agent

Inventors:

Greg I. Chiou, a citizen of the United States, residing at,
19388 Miller Ct.
Saratoga, California 95070

Lev Stesin, a citizen of the United States, residing at,
270 26th Avenue #10
San Francisco, California 94121

Arup Mukherjee, a citizen of Canada, residing at,
110 Harbor Seal Ct.
San Mateo, California 94404

Assignee:

Yahoo! Inc.
3420 Central Expressway
Santa Clara, CA 95051

Entity: Large

TOWNSEND and TOWNSEND and CREW LLP
Two Embarcadero Center, 8th Floor
San Francisco, California 94111-3834
Tel: 415-576-0200

Mobile Software Morphing Agent

COPYRIGHT NOTICE

CROSS-REFERENCES TO RELATED APPLICATIONS

This application is related to U.S. Provisional Patent Application Serial No. 60/212,060 (Atty. Docket No. 017887-005300), filed June 16, 2000, entitled "Mobile Software Morphing Agent," the disclosure of which is hereby incorporated by reference in its entirety.

BACKGROUND OF THE INVENTION

The present invention relates generally to modifying and translating information received from a remote source, and more particularly to modifying and translating executable code and data received from a web site.

The World Wide Web (WWW), or "the Web", is now the premier outlet to publish information of all types and forms. Documents published on the web, commonly called Web pages, are typically published using a markup language such as HTML (or Hyper Text Markup Language), which sets standards for the formatting of documents. Additionally, These standards make it possible for people to read and understand documents no matter which program they use for that purpose. Often included in an HTML formatted web page are software code segments attached as part of the page. Examples of such software include JavaScript, Java and ActiveX commands. If a user's browser is enabled to process the software code, the code will typically be processed to provide additional windows, e.g., pop-up windows, forms and other content for presentation to the user.

Typically, a user accesses pages on the Web through a web portal. One common portal is Yahoo located at URL: http://www.yahoo.com/. When a user selects a reference such as a URL presented on a page provided by the portal, the users browser

will access another page associated with the URL at a remote site. From then on, the user will be connected to the remote server unless the browser is instructed to return to the portal (e.g., via a "back" button or a "home" button displayed on the browser). In the commerce context, for example, a user may access a remote commerce site and conduct transactions, e.g., to purchase a product. In this case, the portal is completely unaware of any transactions or information exchange between the user and the remote site.

It is therefore desirable for a web portal to provide a page from a remote site, such as a remote commerce site, via a special proxy server to a user and to keep the user connected to the proxy so that information exchange between the user and remote server can be monitored by the proxy. However, it may be necessary to modify HTML formatting, HTML links and JavaScript code associated with a page provided by a remote site so that information exchange activity is directed to the proxy. For example, it is desirable to translate a link directed to a particular site into a link directed to the proxy so that the proxy handles access to the desired page from the particular site.

Accordingly, it is desirable to provide a configurable system to parse and translate downloadable software and/or content without additional development efforts from the original software and content provider. Additionally, it is desirable to provide an adaptive system to serve a corresponding software morphing agent to handle the original software and content.

SUMMARY OF THE INVENTION

The present invention provides systems and methods for extending or modifying the behavior of executable code and/or data that can be downloaded to a client device (e.g., a PC, laptop, PalmPilot, PDA or other hand-held communication device, etc.) implementing a browser program (e.g., Microsoft Internet Explorer, Netscape Navigator, a microbrowser such as a WAP enabled browser, etc.). The present invention is particularly useful for modifying web content, such as a web page received from a web site, including JavaScript code and/or HTML data.

According to the invention, one or more morphing agents are provided for translating and modifying code and data from a software source, such as a remote server. Each morphing agent translates and modifies a particular type of code. For example, one morphing agent may be provided for processing JavaScript code and another may be provided for processing HTML code and data. It will be appreciated that one morphing

agent may be provided for processing multiple types of code, for example, one morphing agent for processing JavaScript and HTML code. Each morphing agent typically includes a tokenizer module, a parser module and a translation module, each of which implements specific rule sets. Original software content is first tokenized according to a set of tokenizer rules, and subsequently parsed according to a set of parser rules to determine relationships between the tokens. The parsed code is then translated according to the set of translator rules to produce the desired modified software content. An exception handler module is also provided for implementing exception rules when an exception occurs.

In operation, a user establishes a connection with a proxy server using the browser program on the client device, and the proxy server establishes a connection with the software source. The original software content is downloaded by the proxy server. All modules of a particular morphing agent can be located either on the client device or on the proxy server, or they may be spread between the client device and proxy server. Thus, if all modules reside on the proxy server, the morphing agent modifies the original software content and the modified content is downloaded to the client device. Similarly, if all modules reside on the client device, the original content is downloaded to the client device for processing by the morphing agent at the client device. If some of the modules reside on the proxy server, those module process the original content and the partially processed code is downloaded to the client device for processing by the remaining modules.

According to an aspect of the present invention, a computer implemented method is provided for modifying code to be compatible with a runtime library, wherein the code is received from a remote source. The method typically comprises the steps of receiving the code segment from the remote source, tokenizing the code segment into a plurality of tokens, and parsing the plurality of tokens so as to determine relationships between the plurality of tokens. The method also typically includes the step of translating the code segment into a modified code segment based on the determined relationships between the tokens such that the modified code segment is compatible with the runtime library.

According to another aspect of the present invention, a computer readable medium containing instructions for controlling a computer system to modify a code segment received from a remote source to be compatible with a runtime library is provided. The instructions typically include instructions to tokenize the code segment

into a plurality of tokens, and parse the plurality of tokens so as to determine relationships between the plurality of tokens. The instructions also typically include instructions to translate the code segment into a modified code segment based on the determined relationships between the tokens such that the modified code segment is compatible with the runtime library.

Reference to the remaining portions of the specification, including the drawings and claims, will realize other features and advantages of the present invention. Further features and advantages of the present invention, as well as the structure and operation of various embodiments of the present invention, are described in detail below with respect to the accompanying drawings. In the drawings, like reference numbers indicate identical or functionally similar elements.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 illustrates a general overview of an information retrieval and communication network including a proxy server, client devices, and remote servers according to an embodiment of the present invention;

Figure 2 illustrates various configurations of a runtime environment of a morphing agent according to embodiments of the present invention; and

Figure 3 illustrates a general processing flow between modules of a morphing agent according to an embodiment of the present invention.

DESCRIPTION OF THE SPECIFIC EMBODIMENTS

Figure 1 illustrates a general overview of an information retrieval and communication network 10 including a proxy server 20, clients 30_1 to 30_N, and remote servers 50_1 to 50_N according to an embodiment of the present invention. In computer network 10, clients 30_1 to 30_N are coupled through the Internet 40, or other communication network, to proxy server 20 and servers 50_1 to 50_N. Only one proxy server 20 is shown, but it is understood that more than one proxy server can be used and that other servers providing additional functionality may also be interconnected to any component shown in network 10 either directly, over a LAN or a WAN, or over the Internet.

Several elements in the system shown in Figure 1 are conventional, well-known elements that need not be explained in detail here. For example, each client 30

could be a desktop personal computer, workstation, cellular telephone, personal digital assistant (PDA), laptop, or any other computing device capable of interfacing directly or indirectly to the Internet. Each client 30 typically runs a browsing program, such as Microsoft's Internet Explorer, Netscape Navigator, or a WAP enabled browser in the case of a cell phone, PDA or other wireless device, or the like, allowing a user of client 30 to browse pages available to it from proxy server 20, servers 50_1 to 50_N or other servers over Internet 40. Each client 30 also typically includes one or more user interface devices 32, such as a keyboard, a mouse, touchscreen, pen or the like, for interacting with a graphical user interface (GUI) provided by the browser on a monitor screen, LCD display, etc., in conjunction with pages and forms provided by proxy server 20, servers 50_1 to 50_N or other servers. The present invention is suitable for use with the Internet, which refers to a specific global Internetwork of networks. However, it should be understood that other networks can be used instead of the Internet, such as an intranet, an extranet, a virtual private network (VPN), a non-TCP/IP based network, any LAN or WAN or the like.

According to one embodiment as will be described in more detail below, proxy server 20 and/or clients 30, and all of their related components are operator configurable using an application including computer code run using a central processing unit such as an Intel Pentium processor or the like. Computer code for operating and configuring proxy server 20 and/or clients 30 as described herein is preferably stored on a hard disk, but the entire program code, or portions thereof, may also be stored in any other memory device such as a ROM or RAM, or provided on any media capable of storing program code, such as a compact disk medium, a DVD, a floppy disk, or the like. Additionally, the entire program code, or portions thereof may be downloaded to clients 30 or otherwise transmitted as is well known, e.g., from proxy server 20 over the Internet, or through any other conventional network connection as is well known, e.g., extranet, VPN, LAN, etc., using any communication protocol as is well known.

In general, a user accesses and queries proxy server 20, servers 50_1 to 50_N, and other servers through a client 30 to view and download content such as news stories, advertising content, search query results including links to various websites and so on. Such content can also include other media objects such as video and audio clips, URL links, graphic and text objects such as icons and links, and the like. Additionally, such content is typically presented to the user as a web page formatted according to downloaded JavaScript code and HTML code and data as is well known. It will be appreciated that the techniques of the present invention are equally applicable to

processing other types of code such as Java code and ActiveX code, and any markup language, including any instance of SGML, such as XML, WML, HTML, DHTML and HDML.

According to one embodiment of the invention, a user preferably accesses servers 50_1 to 50_N through proxy server 20. In the context of electronic commerce, for example, a user may access a local commerce site that provides access (via URL or other selectable links or references) to remote commerce sites, such as individual commerce web sites. One such system is described in U.S. Patent Application Serial Number 09/372,350 (Atty. Docket No. 017887-002500), entitled "Electronic Commerce System for Referencing Remote Commerce Sites at a Local Commerce Site," filed August 11, 1999, the contents of which are hereby incorporated by reference in their entirety for all purposes. As described therein, a Remote Merchant Integration Server (RMIS) provides an interface to multiple merchant web sites. A user that accesses a remote commerce site through the RMI proxy is presented with a slightly modified version of the commerce site by the RMI server. Any requests from the user to a remote commerce site is managed by the RMI server and any responses by the remote commerce site are also managed by the RMI server transparently to both the user and the remote commerce site. One advantage of such a system includes the ability to provide, in the commerce context, a single shopping basket to a user who desires to shop at multiple remote commerce sites. Another advantage is the ability to track transactional information associated with users' purchases at the various merchant sites. An example of such a system can be located on the Internet at the Yahoo! Shopping site (URL: http://shopping.yahoo.com/). In this example, it is desirable to modify JavaScript code and HTML code and data received from the remote commerce sites using the techniques of the present invention to facilitate integration of the RMI system and to maintain user connection to the RMI system during transactions with the remote commerce sites.

According to an embodiment of the present invention, a set of different software morphing agents are provided for handling different kinds of software technologies. For example, one morphing agent (MA) is provided for handling JavaScript and another MA is provided for handling HTML. The MA for each type of the original third-party software technology is delivered to the appropriate device(s), e.g., proxy server 20 and/or a client device 30.

Figure 2 illustrates various exemplary configurations of a runtime environment of a morphing agent (MA) according to embodiments of the present

invention. As shown in each configuration, a software source 150, such as a server 50 in Figure 1, provides software to a client device 130 through proxy 120. Depending on the particular MA and its configuration, the software will be modified at the proxy 120, at the client device 130 or partially at the proxy 120 and partially at the client device 130. In configuration a), for example, all components of a particular MA are downloaded to a client device 130 and run in conjunction with a browser application. In configuration b), all components of a particular MA are loaded and run at a proxy server 120. In configuration c), for a particular MA, some components are loaded and run at a proxy server 120 while other components are downloaded to, and run at, a client device 130.

In general, if the code to be modified includes portions that are dynamically assembled, it is preferred that all components for the MA be downloaded to the client device (configuration a). One example of dynamically assembled code in JavaScript could be represented as x + y + "s", where the portion "s" is dynamically assembled or generated by the browser application resident on the client device. Thus, it is preferred that all components of a JavaScript MA be installed on the client side, e.g., at client device 130 to parse and translate dynamically assembled code such as portion "s."

Figure 3 illustrates a general processing flow between modules of a morphing agent according to an embodiment of the present invention. As shown, each morphing agent (MA) includes a tokenizer module 210, a parser module 220 and a translator module 230. Associated with each module is a corresponding rule set, e.g., tokenizer rule set 215, parser rule set 225, and translator rule set 235. An exception handler 240, and associated exception rules set 245, is optionally provided for handling exceptions that occur during the software modification and translation process. Each MA also includes a client-side Runtime Library 250 that includes functions, variable and data configured to run with a browser application. One example of a runtime library can be found in Appendix A.

In operation, before the original software content 260 is processed or executed, all necessary MA components for modifying that particular software type must be installed at the client device 130 (via downloading) and/or at a proxy server 120. The MA then "morphs" (e.g., tokenizes, parses and translates) the original software content 260 into the desired software content 270. Optionally provided exception handler 240 handles any errors that occur during the morphing process. In one embodiment, if an exception, or error, occurs during the morphing process, the exception handler causes the process to be bypassed. The tokenizer 210, parser 220, translator 230, and exception

handler 240 are all configurable via their respective rule sets (i.e. 215, 225, 235, 245). The desired output 270 can then work together with the client-side runtime library 250 (via downloading) in a user's browser.

Once the original software content is received, tokenizer module 210 analyzes the string of characters representing a code segment and maps the characters to various token types according to the tokenizer rule set 215. Typical JavaScript token types include string, integer, keyword, identifier, etc. Parser 220 then parses the resulting set of tokens according to the parser rule set 225 to determine relationships between the token types associated with the code segment being analyzed. In one embodiment, a hierarchical tree structure relating the various tokens is created. One example of tokenizer and parser code useful for tokenizing and parsing JavaScript is the NGS JavaScript Interpreter which can be located and downloaded from the Internet at URL: http://www.ngs.fi/js/, the contents of which are hereby incorporated for reference for all purposes. Translator module 230 then transforms the code segment into the desired software content 270 based on the specific translator rule set 235, the token types, and the relationships between the tokens determined by parser 220.

It may be necessary to modify the above NGS JavaScript Interpreter (tokenizer and parser, in particular) to run more efficiently when integrated with a browser application such as Microsoft Internet Explorer or Netscape Navigator. Appendix B includes examples of a modified tokenizer and parser from NGS JavaScript Interpreter, translator code and code for integrating the modified tokenizer and parser with a Browser, according to one embodiment of the present invention.

A typical translator module (230 + 235) of an MA for transforming JavaScript transforms function calls and variables to new function calls and variables to be used together with a client-side runtime library 250 in a user's browser. For example, a function call "open()" is translated according to one embodiment as follows:

"open (URL, TARGET, OPT)" is translated to

"new_function1(URL, TARGET, OPT)", where the function "new_function1()" is implemented in the client-side runtime library.

Similarly, a variable assignment expression is translated according to one embodiment as follows:

"OBJ.location=URL" is translated to

"OBJ.location=new_function2(URL)", where the function "new_function2" is implemented in the client-side runtime library.

Appendix C illustrates examples of translation rules (e.g., 235) for translating function calls and variables according to one embodiment of the present invention.

While the invention has been described by way of example and in terms of the specific embodiments, it is to be understood that the invention is not limited to the disclosed embodiments. To the contrary, it is intended to cover various modifications and similar arrangements as would be apparent to those skilled in the art. Therefore, the scope of the appended claims should be accorded the broadest interpretation so as to encompass all such modifications and similar arrangements.

WHAT IS CLAIMED IS:

1. A computer implemented method of modifying code to be compatible with a runtime library, wherein the code is received from a remote source, the method comprising the steps of:

receiving a code segment from the remote source;

tokenizing the code segment into a plurality of tokens;

parsing the plurality of tokens so as to determine relationships between the plurality of tokens;

translating the code segment into a modified code segment based on the determined relationships between the tokens such that the modified code segment is compatible with the runtime library.

2. The method of claim 1, wherein the code segment is one of a JavaScript code segment, a Java code segment, an ActiveX code segment and a markup language segment.

3. The method of claim 1, wherein the runtime library is linked to a browser application in a client device communicably coupled to a proxy server, and wherein the steps of receiving, tokenizing, parsing and translating the code segment are performed in the proxy server.

4. The method of claim 3, further including the step of sending the modified code from the proxy server to the client device to be processed by the browser.

5. The method of claim 3, wherein the client device is communicably coupled to the proxy server over the Internet.

6. The method of claim 1, wherein the proxy server performs the steps of receiving, tokenizing, parsing and translating the code segment.

7. The method of claim 1, wherein the runtime library is linked to a browser application in a client device communicably coupled to a proxy server, wherein the step of receiving the code segment from the remote source is performed in the proxy server, wherein the steps of tokenizing, parsing and translating the code segment are performed in the client device, and wherein the method further includes the step of sending the code segment from the proxy server to the client device.

8. The method of claim 7, wherein the code segment includes a dynamically assembled portion.

9. The method of claim 7, wherein the client device is communicably coupled to the proxy server over the Internet.

10. The method of claim 1, wherein the step of translating includes translating a first function call to a second function call, wherein the second function call is compatible with the runtime library.

11. The method of claim 1, wherein the step of translating includes translating a function call to a variable, wherein the variable is compatible with the runtime library.

12. The method of claim 1, wherein the step of translating includes translating a first variable to a second variable, wherein the second variable is compatible with the runtime library.

13. The method of claim 1, wherein the step of translating includes translating a variable to a function call, wherein the function call is compatible with the runtime library.

14. The method of claim 1,

wherein the code segment includes one or more first elements selected from the group consisting of:

digits, characters, keywords, literals, identifiers, operators, expressions, statements, variables, regular expressions, functions, arguments and programs;

wherein the modified code segment includes one or more second elements selected from the group consisting of:

digits, characters, keywords, literals, identifiers, operators, expressions, statements, variables, regular expressions, functions, arguments and programs; and

wherein the second elements are compatible with the runtime library.

15. A computer readable medium containing instructions for controlling a computer system to modify a code segment received from a remote source to be compatible with a runtime library, by:

tokenizing the code segment into a plurality of tokens;

parsing the plurality of tokens so as to determine relationships between the plurality of tokens;

translating the code segment into a modified code segment based on the determined relationships between the tokens such that the modified code segment is compatible with the runtime library.

16. The computer readable medium of claim 15, wherein the code segment is one of a JavaScript code segment, a Java code segment, an ActiveX code segment and a markup language segment.

17. The computer readable medium of claim 15, further comprising instructions for handling an exception when an exception occurs.

18. The computer readable medium of claim 15, wherein the runtime library is implemented on a client device communicably coupled to a proxy server.

19. The computer readable medium of claim 15, wherein the instructions for translating include instructions for translating a function call to a variable, wherein the variable is compatible with the runtime library.

20. The computer readable medium of claim 15, wherein the instructions for translating include instructions for translating a first variable to a second variable, wherein the second variable is compatible with the runtime library.

21. The computer readable medium of claim 15, wherein the instructions for translating include instructions for translating a first function call to a second function call, wherein the second function call is compatible with the runtime library.

22. The computer readable medium of claim 15, wherein the instructions for translating include instructions for translating a variable to a function call, wherein the function call is compatible with the runtime library.

23. The computer readable medium of claim 15, wherein the code segment includes one or more first elements selected from the group consisting of:

digits, characters, keywords, literals, identifiers, operators, expressions, statements, variables, regular expressions, functions, arguments and programs;

wherein the modified code segment includes one or more second elements selected from the group consisting of:

digits, characters, keywords, literals, identifiers, operators, expressions, statements, variables, regular expressions, functions, arguments and programs; and

wherein the second elements are compatible with the runtime library.

MOBILE SOFTWARE MORPHING AGENT

ABSTRACT OF THE DISCLOSURE

Systems and methods for extending or modifying the behavior of mobile (downloadable) software, such as JavaScript, HTML, and/or data that can be downloaded to a client device. One or more morphing agents are provided for translating and modifying code and data from a software source, such as a remote server. Each morphing agent translates and modifies one or more particular types of code. For example, one morphing agent may be provided for processing JavaScript code and another may be provided for processing HTML code and data. Each morphing agent typically includes a tokenizer module, a parser module and a translation module, each of which follows specific rule sets. Original software content is first tokenized according to a set of tokenizer rules, and subsequently parsed according to a set of parser rules. The parsed code is then translated according to the set of translator rules to produce the desired modified software content. An exception handler module is also provided for implementing exception rules when an exception occurs.

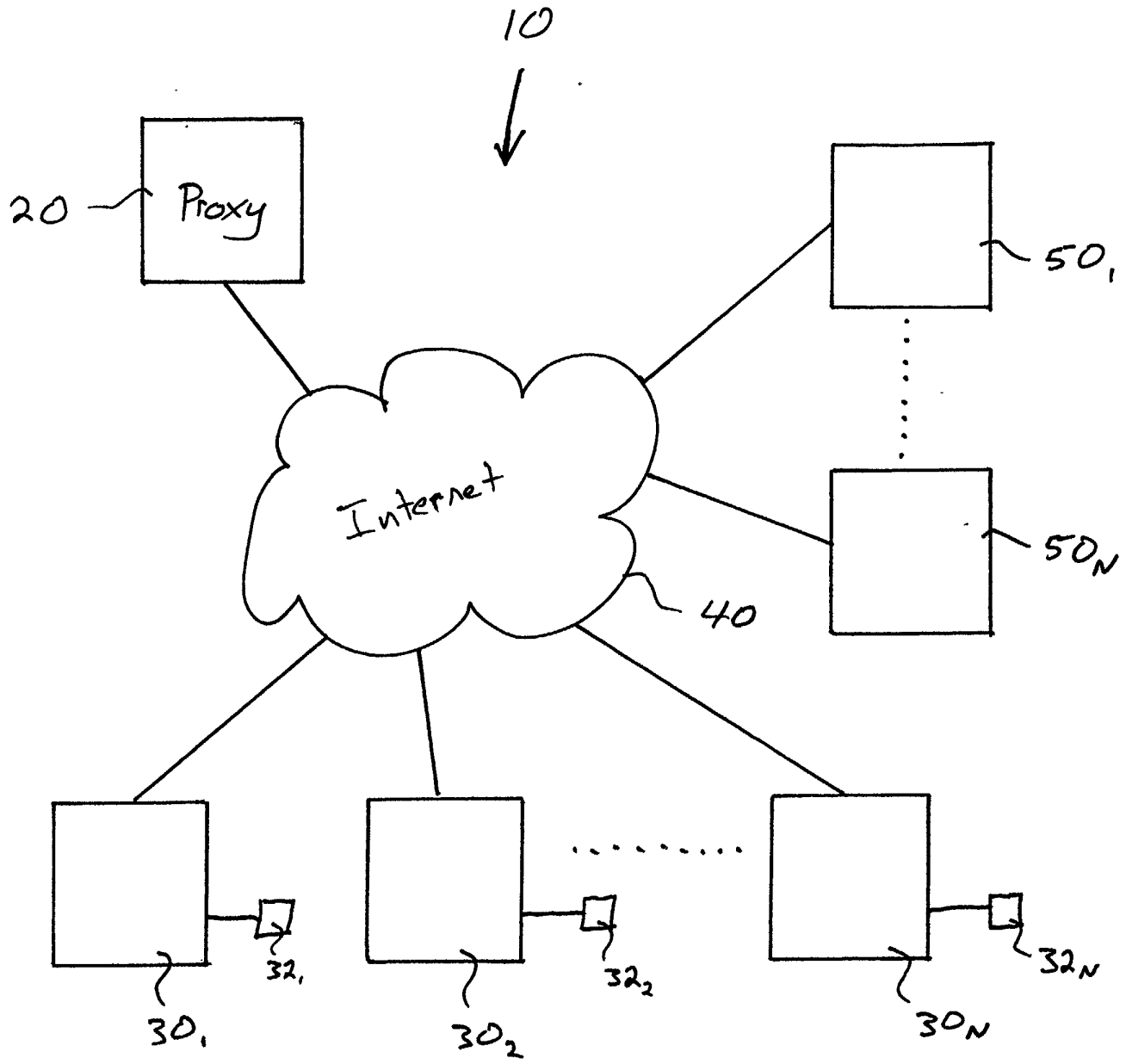

Figure 1

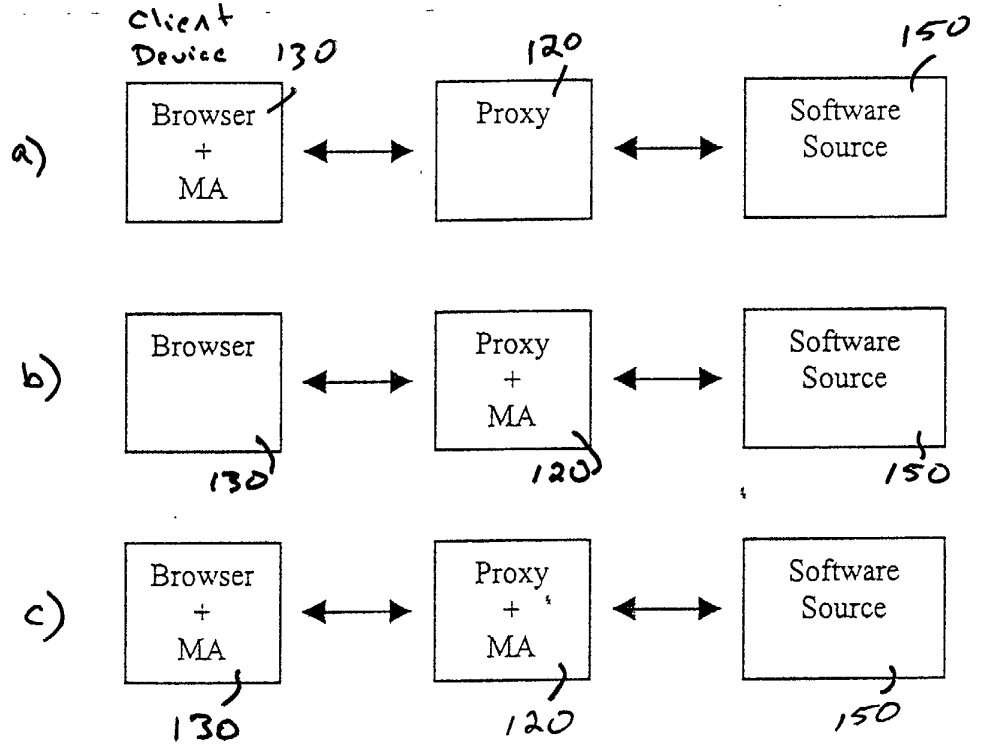

Figure 2

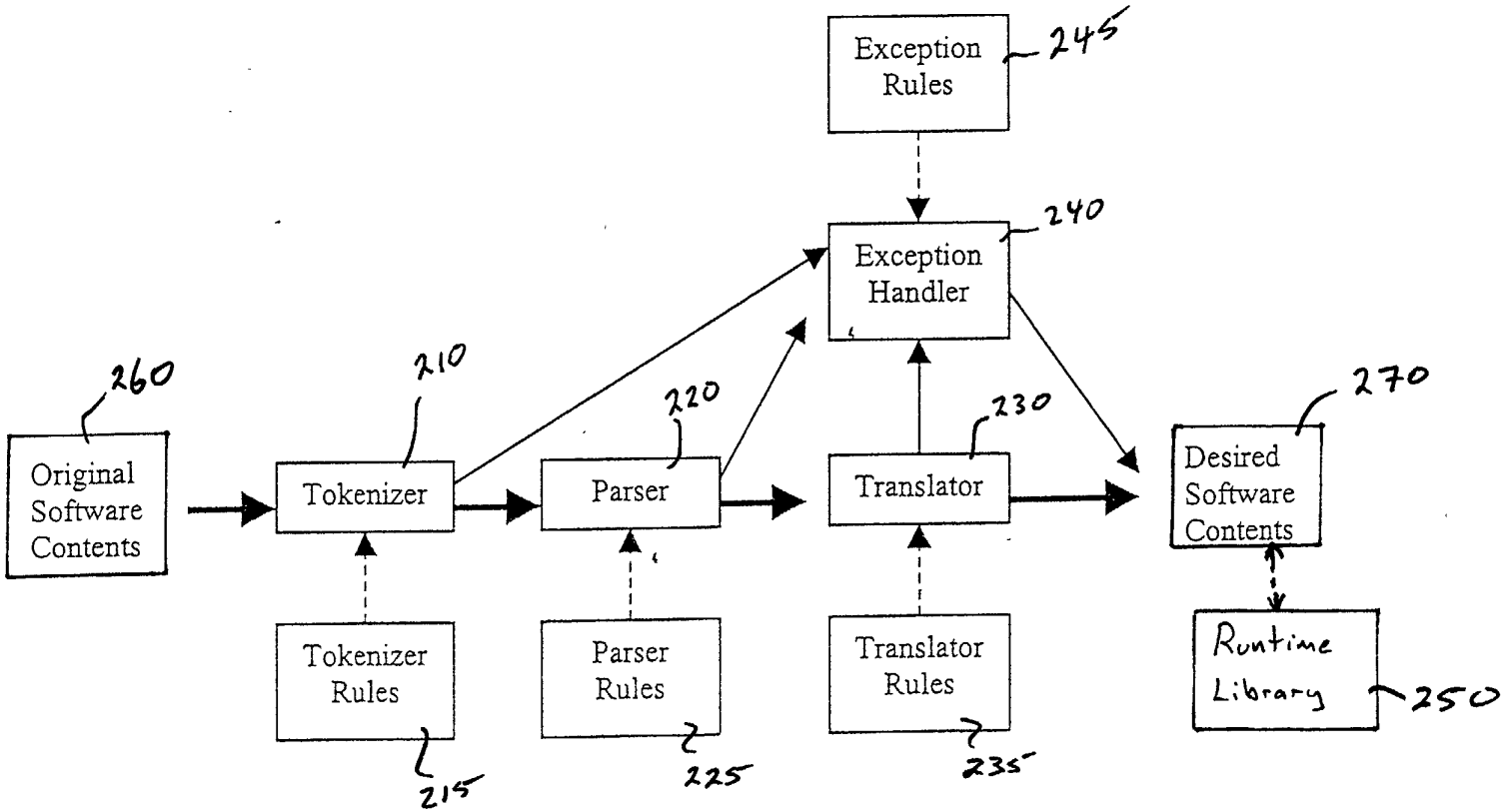

Figure 3

DECLARATION

As a below named inventor, I declare that:

My residence, post office address and citizenship are as stated below next to my name; I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural inventors are named below) of the subject matter which is claimed and for which a patent is sought on the invention entitled: **MOBILE SOFTWARE MORPHING AGENT** the specification of which __X__ is attached hereto or ____ was filed on _____ as Application No. _____ and was amended on _____ (if applicable).

I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above. I acknowledge the duty to disclose information which is material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56. I claim foreign priority benefits under Title 35, United States Code, Section 119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed.

Prior Foreign Application(s)

Country	Application No.	Date of Filing	Priority Claimed Under 35 USC 119

I hereby claim the benefit under Title 35, United States Code § 119(e) of any United States provisional application(s) listed below:

Application No.	Filing Date
60/212,060	June 16, 2000

I claim the benefit under Title 35, United States Code, Section 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, Section 112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, Section 1.56 which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

Application No.	Date of Filing	Status

Full Name of Inventor 1:	Last Name: **CHIOU**	First Name: **GREG**	Middle Name or Initial: **I.**	
Residence & Citizenship:	City: **Saratoga**	State/Foreign Country: **California**	Country of Citizenship: **United States**	
Post Office Address:	Post Office Address: **19388 Miller Court**	City: **Saratoga**	State/Country: **California**	Postal Code: **95070**
Full Name of Inventor 2:	Last Name: **STESIN**	First Name: **LEV**	Middle Name or Initial:	
Residence & Citizenship:	City: **San Francisco**	State/Foreign Country: **California**	Country of Citizenship: **United States**	
Post Office Address:	Post Office Address: **270 26th Ave., #10**	City: **San Francisco**	State/Country: **California**	Postal Code: **94121**

Full Name of Inventor 3:	Last Name: **MUKHERJEE**	First Name: **ARUP**	Middle Name or Initial:	
Residence & Citizenship:	City: **San Mateo**	State/Foreign Country: **California**	Country of Citizenship: **Canada**	
Post Office Address:	Post Office Address: **110 Harbor Seal Court**	City: **San Mateo**	State/Country: **California**	Postal Code: **94404**

I further declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issuing thereon.

Signature of Inventor 1	Signature of Inventor 2	Signature of Inventor 3
Greg I. Chiou	Lev Stesin	Arup Mukherjee
Date	Date	Date

SF 1130552 v1

APPENDIX A

```
<!--

/*************************************************************
 *
 * U.S. Patent Pending. Copyright 1999, 2000 Yahoo! Inc.,
 * 3420 Central Expressway, Santa Clara, California U.S.A.
 *
 * ALL RIGHTS RESERVED.
 *
 * This computer program is protected by copyright law and
 * international treaties. Unauthorized reproduction or
 * distribution of this program, or any portion of it, may
 * result in severe civil and criminal penalties, and will
 * be prosecuted to the maximum extent possible under the law.
 *
 *************************************************************/


/*************************************************************
 * Exception handling
 *************************************************************/
function rmi_handleError (err, url, line)
{
    // alert('BAD: \n' + err + '.\n' + url + '\nline no: ' + line);
    // window.status = 'BAD: \n' + err + '.\n' + url + '\nline no: ' + line;
    window.status = "Javascript: Done (" + line + ")";

    return true;            // error is handled
}
window.onerror = rmi_handleError;


/*************************************************************
 * Globals
 *************************************************************/
var rmi_Vars = "/rmivars%3ftarget=_top";
var rmi_FramesetTagCounter = 0;
var rmi_CookieDomain = ".yahoo.com";

// delete the 1st character, <.
rmi_FrameWrapper = rmi_FrameWrapper.substring(1, rmi_FrameWrapper.length);

/*************************************************************
 * Translate a string, then write to the browser.
 *************************************************************/

function rmi_writeln(obj, str)
{
    var newStr;

    if (arguments.length == 2) {
        newStr = rmi_xlate(str);

        if (obj == document && (typeof document.layers != "undefined")
                && (typeof document.layers['rmilayer'] != "undefined") )
```

```
            document.layers['rmilayer'].document.writeln(newStr);
        else
            obj.writeln(newStr);
    } else {
        newStr = rmi_xlate(obj); // for backward compatibility with hseds
        document.writeln(newStr);
    }
}

function rmi_write(obj, str)
{
    var newStr;

    if (arguments.length == 2) {
        newStr = rmi_xlate(str);

        if (obj == document && (typeof document.layers != "undefined")
                && (typeof document.layers['rmilayer'] != "undefined") )
            document.layers['rmilayer'].document.write(newStr);
        else
            obj.write(newStr);

    } else {
        newStr = rmi_xlate(obj); // for backward compatibility with hseds
        document.write(newStr);
    }
}

/************************************************************
 * String utilities
 ************************************************************/
function rmi_startsWith(full, sub)
{
    var fullLower = full.toLowerCase();
    var subLower = sub.toLowerCase();
    var index = fullLower.indexOf(subLower);
    return index ? false : true;
}

function rmi_endsWith(full, sub)
{
    var fullLower = full.toLowerCase();
    var subLower = sub.toLowerCase();
    var offset = fullLower.length - subLower.length;
    if (offset < 0) return false;

    var index = fullLower.indexOf(subLower, offset);
    return (index==offset) ? true : false;
}

function rmi_endsExactlyWith(full, sub)
{
    var offset = full.length - sub.length;
    if (offset < 0) return false;

    var index = full.indexOf(sub, offset);
    return (index==offset) ? true : false;
```

```
}

/* gets the port of an URL */
function rmi_getPort(url)
{
    var host = rmi_getHost(url);        // get "host:port"
    var begin = host.indexOf(":");

    if (begin == -1 || (host.length - begin) < 2)      // e.g. length of ':80'
    {
        return (rmi_getProtocol(url) == "https") ? "443" : "80" ;
    }
    else
    {
        return host.substring(begin+1, host.length);   // +1 for ':'
    }
}

/* gets the protocol of an URL */
function rmi_getProtocol(url)
{
    var index = url.indexOf("://");
    return url.substring(0, index);
}

/* http://HOST/whatever
 * return HOST
 */
function rmi_getHost(url)
{
    var end = url.indexOf("://");
    var next = end + 3;

    end = url.indexOf("/", next);
    if (end == -1) end = url.length;
    return url.substring(next, end);
}

/* http://HOSTNAME:port/whatever
 * return HOSTNAME
 */
function rmi_getHostname(url)
{
    var host = rmi_getHost(url);        // get "host:port"
    var index = host.indexOf(":");

    if (index == -1)
        return host;
    else
        return host.substring(0, index);
}

/* http://host/FILE
 * return FILE
 */
function rmi_getFile(url)
{
```

```
    var end = url.indexOf("://");
    var next = end + 3;

    end = url.indexOf("/", next);

    if (end == -1)
        return "/";
    else
        return url.substring(end, url.length);
}

/* PROTOCOL://HOST/FILE
 * return URLRoot == PROTOCOL://HOST
 */
function rmi_getURLRoot(url)
{
    var protocol = rmi_getProtocol(url);
    var host = rmi_getHost(url);
    return protocol + "://" + host
}

function rmi_dirname(full)
{
    var dir = full;

    // Remove cgi parameters (e.g. "?k1=v1&k2=v2...")
    // because the parameters might have '/'

    var ind = dir.indexOf('?');
    if ( ind >= 0 ) dir = dir.substring(0, ind);

    ind = dir.lastIndexOf('/');       // search from right
    if (ind == -1) return "";         // no slash
    if (ind == 0)  return "/";        // root

    if ( rmi_endsExactlyWith(dir.substring(0, ind+1), "://") )
        ind = dir.length;

    return dir.substring(0, ind);
}

/* Trim leading & ending quotes
 */
function rmi_trimQuotes(str)
{
    var first = str.charAt(0);
    if (first == '"') return (str.substring(1, str.length-1));
    if (first == '\'') return (str.substring(1, str.length-1));
    return str;
}

/* Trim leading & ending spaces
 */
function rmi_trim(str)
{
    if (typeof str == "undefined") return "";
```

```
    var start = 0;
    var end = str.length;

    for (var i=0; i < str.length; ++i)
    {
        if (str.charAt(i) == ' ') continue;

        start = i;
        break;
    }

    for (var i=str.length-1; i >= 0 ; --i)
    {
        if (str.charAt(i) == ' ') continue;

        end = i + 1;
        break;
    }

    return str.substring(start, end);
}

/************************************************************
 * Normalize URL
 ************************************************************/
function rmi_normalizeURL(in_url)
{
    var url = in_url.toString();
    var first = url.charAt(0);
    if (first == '"') url = url.substring(1, url.length-1);

    var ret = url;

    if ( rmi_startsWith(url, "http://") )
    {
        ret = url;
    }
    else if ( rmi_startsWith(url, "https://") )
    {
        ret = url;
    }
    else if ( rmi_startsWith(url, "/") )
    {
        ret = rmi_getURLRoot(rmi_BaseURL) + url;
    }
    else if ( rmi_startsWith(url, "#") )
    {
        ret = document.location + url;
    }
    else        // relative
    {
        var dir = rmi_dirname(rmi_BaseURL);
        ret = dir + "/" + url;
    }

    return ret;
}
```

```
/*************************************************************
 * Translate a URL (in the case of form action)
 * If the incoming code is the form of location=url
 *   then we return location=rmi_xlateURL(url)
 *************************************************************/
function rmi_xlateAction(action_url)
{
    var ret = "";
    var url = action_url.toString();

    if ( rmi_startsWith(url, "location=") ) {
       var new_loc = url.substring(url.indexOf("=")+1, url.length);
       ret = "location='" + rmi_xlateURL(new_loc) + "'";
    } else {
       ret = rmi_xlateURL(url);
    }

    if (rmi_JsDebug.indexOf(",rmi_xlateAction,") != -1)
        alert("rmi_xlateAction: old: " + action_url + "\n" + "new: " + ret);

    return ( ret );
}

/*************************************************************
 * Translate a URL
 * Note: if url is already starts with rmi proxy url it will
 *       not be translated again.
 *************************************************************/
function rmi_xlateURL(in_url)
{

    var ret = "";
    var url = in_url.toString();
    var first = url.charAt(0);
    if (first == '"') url = url.substring(1, url.length-1);
    if (first == '\'') url = url.substring(1, url.length-1);

    // Ignore javascript:

    if ( rmi_startsWith(url, "javascript:") )
        return url;

    url = rmi_normalizeURL(url);

    if (  rmi_startsWith(url, rmi_ProxyURL) ||
          rmi_startsWith(url, rmi_SecureProxyURL) ||
          rmi_endsWith(url, ".jpg") ||
          rmi_endsWith(url, ".jpeg") ||
          rmi_endsWith(url, ".gif")
       )
    {
        return url;
    }

    /*
     *  Collapse the file part of an URL
```

```
     */
    var urlroot = rmi_getURLRoot(url);
    var file = pathCollapse(rmi_getFile(url));

    ret = urlroot + file;

    if ( rmi_startsWith(url, "https://"))
        ret = rmi_SecureProxyURL + ret;
    else
        ret = rmi_ProxyURL + ret;

    if (rmi_FrameWrapperMode && rmi_UrlTarget == "_top")    // onTop & wrapper
mode
        ret = rmi_appendToUrl(ret, rmi_Vars);

    if (rmi_JsDebug.indexOf(",rmi_xlateURL,") != -1)
        alert("rmi_xlateURL:\n" + "in_url: " + in_url + "\n" + "ret: " + ret +
"\n");

    return ret;
}

/************************************************************
 * Get original (before RMI) location property (href, host, etc)
 ************************************************************/
function rmi_getOriginal(loc, prop)
{
    var origUrl = "" + loc;
    var url = "" + loc;
    var index = url.indexOf("/rmi/");
    var ret = "";

    if (index != -1)
        origUrl = url.substring(index+5, url.length);    // Get string after
"/rmi/"

    if (prop == "host")
        ret = rmi_getHostname(origUrl) + ":" + rmi_getPort(origUrl);
    else if (prop == "hostname")
        ret = rmi_getHostname(origUrl);
    else if (prop == "port")
        ret = rmi_getPort(origUrl);
    else if (prop == "pathname")
    {
        var path = rmi_getFile(origUrl);
        ret = (path.indexOf("?") == -1 ) ? path : path.substring(0,
path.indexOf("?"));
    }
    else if (prop == "search")
    {
        var path = rmi_getFile(origUrl);
        ret = (path.indexOf("?") == -1) ? "" : path.substring(path.indexOf("?"),
path.length);
    }
    else                          // location, location.href, & all others
        ret = origUrl;
```

```
    // Remove RMI var string (e.g. /rmivars%3f...).
    // KEEP text before AND after RMI var strings

    var rmiVarStr = "/rmivars";
    var rmiVarStrLen = 9;          // length of "/rmivars?" for IE

    var i_rmiVarStr = ret.indexOf(rmiVarStr);
    var head = (i_rmiVarStr == -1) ? ret : ret.substring(0, i_rmiVarStr);

    var tail = "";
    if (i_rmiVarStr != -1)           // RMI var string exists
    {
        var i1 = ret.indexOf("?", i_rmiVarStr + rmiVarStrLen);
        var i2 = ret.indexOf("#", i_rmiVarStr + rmiVarStrLen);

        if (i1 != -1) tail = ret.substring(i1, ret.length);
        else if (i2 != -1) tail = ret.substring(i2, ret.length);
    }

    return head + tail;
}

/*************************************************************
 * Get original (before RMI) document.domain property
 *************************************************************/
function rmi_getOriginalDomain()
{
    var origUrl = "" + window.location;
    var url = origUrl;
    var index = url.indexOf("/rmi/");
    var ret = "";

    if (index != -1)
        origUrl = url.substring(index+5, url.length);    // Get string after
"/rmi/"

    ret = rmi_getHostname(origUrl);

    // Remove RMI tail string (e.g. /rmivars%3f...)

    var rmiTail = "/rmivars%3f";
    ret = (ret.indexOf(rmiTail) == -1) ? ret : ret.substring(0,
ret.indexOf(rmiTail));

    return ret;
}

/*************************************************************
 * Return a frame object
 *************************************************************/
function rmi_getFrame(win, index)
{
    if (!rmi_FrameWrapperMode) return (win.frames[index]);

    // FrameWrapperMode from here on...
```

```
    if ((typeof index) == "number")
    {
        if (win == top)
            return(win.frames[index+1]);     // +1 due to Yahoo's extra frame
        else
            return(win.frames[index]);
    }
    else                                     // string & other types
        return(win.frames[index]);
}

/************************************************************
 * Get the current dimension of the RMI bar
 *
 * h = rmi_getBarDimensions("height")
 *
 * <FUTURE> w = rmi_getBarDimensions("width")
 *
 ************************************************************/
function rmi_getBarDimensions(dimType)
{
    var doc;

    var defaultRet = 50;

    if (top.frames.length == 0)      // non-frame site
        doc = document;
    else
    {

        //   doc = top.frames[0].document;

        return defaultRet;
    }

    if ( window.navigator.appName.toLowerCase().indexOf("microsoft") != -1 )
    {
        // IE

        if (typeof doc.all.rmi_south_gif == "undefined")
            return defaultRet;
        else
        {
            if (dimType == "height")
                return doc.all.rmi_south_gif.offsetTop;
            else
                return defaultRet;
        }
    }
    else
    {
        // netscape

        if (typeof doc.rmi_south_gif == "undefined")
            return defaultRet;
        else
        {
```

```
            if (dimType == "height")
                return doc.rmi_south_gif.y;
            else
                return defaultRet;
        }
    }

    return defaultRet;       // no match (shouldn't be here)
}

/************************************************************
 * Translate the options before opening a window (e.g. window.open)
 ************************************************************/
function rmi_xlateWinOpt(options)
{
    var tokens = options.split(",");

    var ret = "";
    for (var i=0; i<tokens.length; ++i)
    {
        var pair = tokens[i].split("=");
        var key = rmi_trim(pair[0]);
        var val = rmi_trim(pair[1]);

        if (key == "height")
        {
            var offset = rmi_getBarDimensions(key);

            if (val == "") val = "0";               // if no height value!
            val = "" + (parseInt(val) + offset);
        }

        if (i != 0) ret += ",";

        if (val == "")                  // if no value
            ret += key;
        else
            ret += key + "=" + val;
    }

    return ret;
}

/************************************************************
 * Open a window (using a window object from 1st argument)
 ************************************************************/
function rmi_winobj_open(winobj, url, target, options)
{
    //alert(url);
    //alert(target);
    //alert(options);

    var win;

    if (arguments.length == 2)
        win = winobj.open(rmi_xlateURL(url));
    else if (arguments.length == 3)
```

```
        win = winobj.open(rmi_xlateURL(url), target);
    else
        win = winobj.open(rmi_xlateURL(url), target, rmi_xlateWinOpt(options) );

    if (win != null) win.opener = self;

    return win;
}

/************************************************************
 * Open a window (using a default window object)
 ************************************************************/
function rmi_window_open(url, target, options)
{
    //alert(url);
    //alert(target);
    //alert(options);

    var win;

    if (arguments.length == 1)
        win = window.open(rmi_xlateURL(url));
    else if (arguments.length == 2)
        win = window.open(rmi_xlateURL(url), target);
    else
        win = window.open(rmi_xlateURL(url), target, rmi_xlateWinOpt(options) );

    if (win != null) win.opener = self;

    return win;
}

function rmi_window_open_self(url)
{
    return window.open(rmi_xlateURL(url), "_self");
}

/************************************************************
 * Get 'top' window for RMI
 ************************************************************/
function rmi_getTop(win)
{
    if (rmi_FrameWrapperMode)                                    // frame wrapper
mode
        return win;
    else if (top.frames.length > 1)                              // old frame
mode
        return (win == top) ? top._rmi_bottom : win;
    else                                                         // non-frame
mode
        return win;
}

/************************************************************
 * Translate a target URL, then replace the document in
 * the target window
 ************************************************************/
```

```
function rmi_replace(win, url)
{
    if (win == "") win = self;

    if (rmi_FrameWrapperMode)
    {
        if (win == top)
            win.location.replace(rmi_xlateURL(url) + rmi_Vars);
        else
            win.location.replace(rmi_xlateURL(url));
    }
    else if (top.frames.length > 1)            // old frame mode
    {
        if (win == top)
            top._rmi_bottom.location.replace(rmi_xlateURL(url));
        else
            win.location.replace(rmi_xlateURL(url));
    }
    else                                                  // non-frame mode
        win.location.replace(rmi_xlateURL(url));
}

/*************************************************************
 * Handle location setting for different modes after JS translation
 *
 * <Sample translation>
 * From: window.top.parent.location.href = url;
 *   To: rmi_setLocation("window.top", ".parent.location.href",
rmi_xlateURL(url), window.top.parent);
 *************************************************************/
function rmi_setLocation(s1, s2, url, win)
{
    var frameName = "";
    var newUrl = url;

    if (rmi_FrameWrapperMode)
    {
        //@@ if (rmi_startsWith(s2, ". location"))
        if ( win == top )
        {
                frameName = "";
                newUrl =  rmi_appendToUrl(url, rmi_Vars);
        }

        // Handle topmost frames

        var aWin = eval(s1);
        var array = s2.split(".");
        var head = rmi_trim(array[0]);

        if (aWin == top && rmi_startsWith(head, "frames" ))
        {
            var i0 = head.indexOf("[");
            var i1 = head.indexOf("]");

            var num = 0;
            num = head.substring(i0+1, i1);
```

```
            if (num >= 0)     // If a valid frame number, increment it
            {
                array[0] = "frames[" + (++num) + "]";
                s2 = array.join(".");
            }
        }
    }
    else                                // old mode
    {
        frameName = "._rmi_bottom";
        newUrl = url
    }

    var code = s1 + frameName + "." + s2 + " = \"" + newUrl + "\";";
    eval(code);

    if (rmi_JsDebug.indexOf(",rmi_setLocation,") != -1)
        alert("rmi_setLocation:\n" + "url: " + url + "\n" + "code: " + code +
"\n");
}

/************************************************************
 * Xlate a String
 ************************************************************/

/*
 *   * If it returns a value different from str
 *     rmi_xlate will return new value.
 * else    (i.e. rmi_xlate_merchant returns str)
 *     rmi_xlate will do regular processing
 */
function rmi_xlate_merchant(str)
{
    // alert("merchant dummy function");
    return str;
}



function rmi_xlate(pStr)
{
    var xlatedStr = "";
    var iSearch, iFrame, iImg, length, startLoc, endLoc;
    var offset1, offset2, head, src, tail;
    var str = "" + pStr;                    // to string to be sure
    var lowercaseStr = str.toLowerCase();

    //
    // invoke merchant specific stuff
    //
    xlatedStr = rmi_xlate_merchant(str);
    if (xlatedStr != str) return xlatedStr;

    var parseStr = rmi_parseloop(str);
    if (parseStr != str) return parseStr;
```

```
    // debugging block begin ////////////////////////////////////
    //xlatedStr = rmi_xlate_src_href(str);
    //if (parseStr != str) {
    //  if (parseStr != xlatedStr) {
    //    alert("parseStr " + parseStr + "\n  xlatedStr " + xlatedStr);
    //  }
    //  return parseStr;
    //}
    // debugging block end ////////////////////////////////////

    return(str);
}

function rmi_parseloop(str)
{
    var tagStr = str;
    var newStr = "";

    while (1) {
        var left, tag, right, nexttag;
        var l, r;

        l = tagStr.indexOf("<");

        // if there is no "<" sign, return tagStr
        if (l == -1) {
            newStr = newStr + tagStr;
            //alert("no < found in " + tagStr + "\n" + newStr);
            break;
        }
        left = tagStr.substring(0, l+1);

        r = tagStr.indexOf(">");
        // if there is no ">" sign, return tagStr
        if (r == -1) {
            newStr = newStr + tagStr;
            //alert("NO > found in " + tagStr + "\n" + newStr);
            break;
        }
        tag = tagStr.substring(l+1, r);

        nexttag = tagStr.indexOf("<", r);

        if (r < l) {
            // if " ... > .. <...>", then add upto < and
            // then loop back
            newStr = newStr + left;
            tagStr = tagStr.substring(l+1, tagStr.length);
        } else if (nexttag == -1) {
            right = tagStr.substring(r, tagStr.length);

            tag = rmi_xlate_src_href(tag);
            tag = rmi_xlate_form_action(tag);
            tag = rmi_xlate_frameset(tag);      // do frameset last because
extra tags are added

            if (rmi_FrameWrapperMode)
```

```
                tag = rmi_doTargetInFrameWrapperMode(tag);
            else
                tag = rmi_xlate_target(tag);

            newStr = newStr + left + tag + right;
            break;
        } else {
            right = tagStr.substring(r, nexttag);

            tag = rmi_xlate_src_href(tag);
            tag = rmi_xlate_form_action(tag);
            tag = rmi_xlate_frameset(tag);      // do frameset last because
extra tags are added

            if (rmi_FrameWrapperMode)
                tag = rmi_doTargetInFrameWrapperMode(tag);
            else
                tag = rmi_xlate_target(tag);

            newStr = newStr + left + tag + right;
            tagStr = tagStr.substring(nexttag, tagStr.length);

            // newStr = newStr + "_" + left + "#" + tag + "#" + right + "_";
            // loop back
        }
    }


    if (str != "" && newStr == "") {
        newStr = str;
    }



    if (rmi_JsDebug.indexOf(",rmi_parseloop,") != -1)
        alert("parseloop:\n" + "old: " + str + "\n" + "new: " + newStr);


    // debugging block begin ////////////////////////////////////
    //var lowercaseStr = str.toLowerCase();
    //if ((lowercaseStr.indexOf("src=") != -1 ||
    //    lowercaseStr.indexOf("href=") != -1)
    //    && lowercaseStr.indexOf("image ") == -1)
    //    {
    //        alert("orig " + str + "\npars " + newStr);
    //    }
    // debugging block end ////////////////////////////////////

    return newStr;
}

function rmi_xlate_src_href(str)
{
    var newStr = "";
    var iSearch, iFrame, iImg, length, startLoc, endLoc;
    var offset1, offset2, head, src, tail;
    var lowercaseStr = str.toLowerCase();
```

```
    iSearch = lowercaseStr.indexOf("src=");
    if (iSearch != -1) {
        length = 4;  // length of "src="

        // should not contain IMAGE tag
        iImg = lowercaseStr.indexOf("image ");
        if (iImg < iSearch && iImg > -1) return str;

        // should not contain IMG tag
        iImg = lowercaseStr.indexOf("img ");
        if (iImg < iSearch && iImg > -1) return str;

        iFrame = lowercaseStr.indexOf("frame");
        if (iFrame == -1) return str;

    } else {
        iSearch = lowercaseStr.indexOf("href=");
        if (iSearch == -1) return str;

        // alert("found href in " + str);
        length = 5;   // length of "href="
    }

    startLoc = iSearch + length;
    head = str.substring(0, startLoc);


    offset1 = lowercaseStr.indexOf(" ", startLoc);
    if (offset1 == -1) {
        offset2 = lowercaseStr.indexOf(">", startLoc);
        if (offset2 == -1) {
            endLoc = str.length;
        } else {
            endLoc = offset2;
        }
    } else {
        endLoc = offset1;
    }

    src = str.substring(startLoc, endLoc);
    tail = str.substring(endLoc, str.length);

    // Ignore 'javascript:*' & RETURN original string

    if ( rmi_startsWith(src.toLowerCase(), "'javascript:") ) return (str);
    if ( rmi_startsWith(src.toLowerCase(), "javascript:") ) return (str);

    var saved_urlTarget = rmi_UrlTarget;            // saved to be restored
later
    if (head.toLowerCase().indexOf("frame ") != -1)
        rmi_UrlTarget = "";                         // <frame> will not be on
top

    newStr = head + rmi_xlateURL(src) + tail;

    rmi_UrlTarget = saved_urlTarget;                // restore it
```

```
    if (rmi_JsDebug.indexOf(",rmi_xlate_src_href,") != -1)
        alert("rmi_xlate_src_href\n" + "old: " + str + "\n" + "new: " + newStr);

    return newStr;
}

function rmi_xlate_form_action(str)
{
    var lowercaseStr = str.toLowerCase();
    var iForm = lowercaseStr.indexOf("form");

    if (iForm == -1) return str;
    // alert (str);

    var iSearch = lowercaseStr.indexOf("action=");
    var length  = 7; // length of "action="
    if (iSearch == -1) {
        iSearch = lowercaseStr.indexOf("action =");
        length = 9; // one more than length of string, to allow for extra space
    }
    if (iSearch == -1) return str;

    var startLoc, endLoc, offset1, offset2, head, src, tail;

    startLoc = iSearch + length;
    head = str.substring(0, startLoc);

    offset1 = lowercaseStr.indexOf(" ", startLoc);
    if (offset1 == -1) {
        offset2 = lowercaseStr.indexOf(">", startLoc);
        if (offset2 == -1) {
            endLoc = str.length;
        } else {
            endLoc = offset2;
        }
    } else {
        endLoc = offset1;
    }

    src = str.substring(startLoc, endLoc);
    tail = str.substring(endLoc, str.length);

    newStr = head + rmi_xlateURL(src) + tail;

    // alert(newStr);

    return newStr;
}

/*************************************************************
 * Write out the 'frame wrapper'
 *************************************************************/
function rmi_writeFrameWrapper()
{
    document.write(rmi_FrameWrapperText);
}
```

```
/************************************************************
 * Handle a frameset tag (for top window in FrameWrapperMode ONLY)
 ************************************************************/
function rmi_xlate_frameset(tag)
{
    if (! rmi_FrameWrapperMode) return tag;
    if (self != top) return tag;

    var lowercaseStr = tag.toLowerCase();
    var iOpenTag = lowercaseStr.indexOf("frameset ");
    var iClosingTag = lowercaseStr.indexOf("/frameset");

    if (iOpenTag == -1 && iClosingTag == -1) return tag;    // not frameset tag

    if (iClosingTag >= 0)       // see </frameset>
    {
        -- rmi_FramesetTagCounter;

        // add Yahoo's </frameset> after the last frameset

        if (rmi_FramesetTagCounter == 0)
            ret = tag + ">/n</frameset";
        else
            ret = tag;
    }
    else                        // see <frameset>
    {
        // add Yahoo's <frameset> before the 1st frameset

        if (rmi_FramesetTagCounter == 0)
            ret = rmi_FrameWrapper + "\n<" + tag;
        else
            ret = tag;

        ++ rmi_FramesetTagCounter;     // count <frameset> tag
    }

    if (rmi_JsDebug.indexOf(",rmi_xlate_frameset,") != -1)
        alert("rmi_xlate_frameset:\n" + "old: " + tag + "\n" + "new: " + ret);

    return (ret);
}

function rmi_xlate_target(str)
{
    var newStr = "";
    var iSearch, iFrame, iImg, length, startLoc, endLoc;
    var offset1, offset2, head, src, tail;
    var lowercaseStr = str.toLowerCase();
    var loc1, loc2, loc3;

    if (rmi_merchant_frames != "yes") {
        // alert("merchant frames not yes");
        return str;
    }
```

```
    loc1 = lowercaseStr.indexOf("target=\"_top\"");
    loc2 = lowercaseStr.indexOf("target=_top");
    loc3 = lowercaseStr.indexOf("target='_top'");

    if (loc1 != -1) {
       iSearch = loc1;
       length = 13;              // length of target="_top"
    } else if (loc2 != -1) {
       iSearch = loc2;
       length = 11;              // length of target=_top
    } else if (loc3 != -1) {
       iSearch = loc3;
       length = 13;              // length of target='_top'
    } else {
       return str;
    }

    startLoc = iSearch;
    endLoc   = startLoc + length;

    head = str.substring(0, startLoc);
    src = "target=\"_rmi_bottom\"";
    tail = str.substring(endLoc, str.length);

    newStr = head + src + tail;

    // alert("head " + head + "\nsrc= " + src + "\ntail " + tail);
    // alert("str= " + str + "\nnew= " + newStr);

    return newStr;
}

/************************************************************
 * Get a attribute value in a tag
 ************************************************************/
function rmi_getTagAttribute(tag, key)
{
    var loc1 = tag.toLowerCase().indexOf(key);
    var loc2 = tag.indexOf("=", loc1) + 1;      // plus 1 for "="
    var first = loc2;
    var last = tag.length;

    if (loc1 == -1) return "";

    var whitespace_trimmed = false;
    for (var i=loc2; i<tag.length; ++i)
    {
        var aChar = tag.charAt(i);

        if (aChar != ' ' && ! whitespace_trimmed)
        {
            first = i;
            whitespace_trimmed = true;
        }

        if (aChar == ' ')
        {
```

```
                last=i;
                if (whitespace_trimmed) break
            }
        }

        if (first == -1)
            retTag = "";
        else
            retTag = tag.substring(first, last);

        if (rmi_JsDebug.indexOf(",rmi_getTagAttribute,") != -1)
        {
            var msg =  "key: " + key + "\n";
                msg += "old: " + tag + "\n";
                msg += "ret: " + retTag + "\n";
                msg += "first: " + first + "\n";
                msg += "last: " + last + "\n";
            alert("rmi_getTagAttribute:\n" + msg);
        }

        return retTag;
    }

    /************************************************************
     * Set a new attribute value in a tag
     ************************************************************/
    function rmi_setTagAttribute(tag, key, newval)
    {
        var loc1 = tag.toLowerCase().indexOf(key);
        var loc2 = tag.indexOf("=", loc1) + 1;      // plus 1 for "="
        var first = loc2;
        var last = tag.length;

        if (loc1 == -1) return tag;

        var whitespace_trimmed = false;
        for (var i=loc2; i<tag.length; ++i)
        {
            var aChar = tag.charAt(i);

            if (aChar != ' ' && ! whitespace_trimmed)
            {
                first = i;
                whitespace_trimmed = true;
            }

            if (aChar == ' ')
            {
                last=i;
                if (whitespace_trimmed) break
            }
        }

        if (first == -1)
            retTag = tag;
        else
```

```
        retTag = tag.substring(0, first) + newval + tag.substring(last,
tag.length);

    if (rmi_JsDebug.indexOf(",rmi_setTagAttribute,") != -1)
    {
        var msg =  "key: " + key + "\n";
            msg += "newval: " + newval + "\n";
            msg += "old: " + tag + "\n";
            msg += "new: " + retTag + "\n";
        alert("rmi_setTagAttribute:\n" + msg);
    }

    return retTag;
}

/************************************************************
 * Handle the target within a tag in a frame wrapper mode
 ************************************************************/
function rmi_doTargetInFrameWrapperMode(tagStr)
{
    if (! rmi_FrameWrapperMode ) return tagStr;


    var retTag = tagStr;

    // ignore frames (will not be on top)

    if (rmi_startsWith(tagStr.toLowerCase(), "frame")) return retTag;

    if (rmi_UrlTarget == "_top")       // onTop & wrapper mode - force to encode
ALL
    {
        retTag = rmi_encodeTarget(tagStr, "href", true);
        retTag = rmi_encodeTarget(retTag, "action", true);
    }
    else            // encode only if target==_top
    {
        retTag = rmi_encodeTarget(tagStr, "href", false);
        retTag = rmi_encodeTarget(retTag, "action", false);
    }

    if (rmi_JsDebug.indexOf(",rmi_doTargetInFrameWrapperMode,") != -1)
        alert("rmi_doTargetInFrameWrapperMode:\n" + "old: " + tagStr + "\n" +
"new: " + retTag);

    return retTag;
}

/************************************************************
 * Encode a target into a URL within a tag
 ************************************************************/
function rmi_encodeTarget(tagStr, key, force)
{
    var retTag = tagStr;

    var oldUrl = rmi_getTagAttribute(tagStr, key);
```

```
    if ( rmi_endsExactlyWith(oldUrl, rmi_Vars) )        // already encoded
        return retTag

    if (! rmi_startsWith(oldUrl, rmi_ProxyURL) &&
        ! rmi_startsWith(oldUrl, rmi_SecureProxyURL) )
        return retTag                                   // not translated (e.g.
gif)

    if (force)      // force to rewrite
    {
        retTag = rmi_setTagAttribute(tagStr, key, oldUrl + rmi_Vars);
    }
    else
    {
        // If target==_top
        var targetVal = rmi_getTagAttribute(tagStr, "target");
        targetVal = rmi_trimQuotes(targetVal);

        if (targetVal == "_top")
        {
            retTag = rmi_setTagAttribute(tagStr, key, oldUrl + rmi_Vars);
        }
    }

    if (rmi_JsDebug.indexOf(",rmi_encodeTarget,") != -1)
        alert("rmi_encodeTarget\n" + "old: " + tagStr + "\n" + "new: " +
retTag);

    return retTag;
}

/*************************************************************
 * Append a string to a URL if no such string at the end yet
 *************************************************************/
function rmi_appendToUrl(url, str)
{
    var urlStr = "" + url;
    var ret;

    if ( rmi_endsExactlyWith(urlStr, str) )                    // See
/rmivars%3f...
        ret = urlStr;
    else if ( rmi_endsExactlyWith(urlStr, unescape(str) ) )    // See
/rmivars?...
    {
        var array = urlStr.split(unescape(str));
        ret = array[0] + str;
    }
    else
        ret = urlStr + str;

    return ret;

}

/*************************************************************
 * Collapse a path (i.e. remove parts of a path like "dir/..")
```

```
 ************************************************************/
function pathCollapse(path)
{
    var slist = path.split("/");
    var stack = new Array();
    var counter = 0;

    for (var i = 1; i < slist.length; ++i)
    {
        var item = slist[i];

        if (item != "..")
            stack[counter++] = item;
        else if (counter > 0)
            --counter;
    }

    stack.length = counter;

    //alert("mpath " + path + "\nmpath " + "/" + stack.join("/"));
    return ("/" + stack.join("/"));
}

/************************************************************
 * Translate a string, then do eval().
 ************************************************************/
function rmi_eval(code)
{
    return eval(code);
}

/************************************************************
 * This function will be overriden at run time if necessary
 ************************************************************/
function rmi_xjs(code)
{
    return code;
}

/************************************************************
 * Translate a string, then do setTimeout().
 ************************************************************/
function rmi_setTimeout(code, msec)
{
    return setTimeout(code, msec);
}

/************************************************************
 * Get RMI cookies
 ************************************************************/
function rmi_getCookie(cookie)
{
    // alert("rmi_getCookie:\n" + "cookies: " + cookie + "\nrmi_cookies: " +
rmi_CurrentCookies);

    if (typeof rmi_CurrentCookies == "undefined")
        return "";
```

```
    else
        return rmi_CurrentCookies;
}

/************************************************************
 * Set RMI cookies
 ************************************************************/
function rmi_setCookie(cookieLHS, cookieRHS)
{
    // Set RMI cookie @ the server side

    var serverCookie = rmi_xlateServerCookie( cookieRHS );
    if (serverCookie == "") return;

    var newCookieTail = "path=/rmi; domain=" + rmi_CookieDomain;
    var newCookie = "rmiCookie" + (new Date()).getTime() + "=" +
escape(serverCookie) + "; " + newCookieTail;

    document.cookie = newCookie;

    // Set rmi_CurrentCookies @ the client side

    var clientCookie = rmi_xlateClientCookie( cookieRHS );
    if (clientCookie == "") return;

    if (typeof rmi_CurrentCookies == "undefined")
        rmi_CurrentCookies = clientCookie;
    else
        rmi_CurrentCookies += ";" + clientCookie;
}

/************************************************************
 * Verify cookie's domain (before setting the cookie)
 ************************************************************/
function rmi_verifyCookieDomain(domain)
{
    var hostname = rmi_getOriginal(window.location, 'hostname');

    if (rmi_endsExactlyWith(hostname.toLowerCase(), domain.toLowerCase()) )
        return true;
    else
        return false;
}

/************************************************************
 * Parse a cookie string, returns a new client cookie
 * (for browsers) without path, domain, expires, & secure fields.
 ************************************************************/
function rmi_xlateClientCookie(cookieStr)
{
    var list = cookieStr.split(";");
    var ret = "";

    for (var i = 0; i < list.length; ++i)
    {
        // NOTE: array's length > 2 if there are more than 2 '='
```

```
        var array = list[i].split("=");

        if (array.length < 1) continue;

        var key = rmi_trim(array[0]).toLowerCase();

        // Verify the cookie domain if there

        if (key == "domain")
        {
            var domainVal = rmi_trim(array[1]).toLowerCase();
            if ( rmi_verifyCookieDomain(domainVal) )
            {
                continue;               // OK
            }
            else
            {
                ret = "";
                break;
            }
        }

        if (key == "path") continue;
        if (key == "expires") continue;
        if (key == "secure") continue;

        if (i != 0) ret += ";"
        ret += array.join("=")
    }

    return ret;
}

/*************************************************************
 * Parse a cookie string, returns a new server cookie
 * (for rmi proxy server) with path & domain (if not there originally).
 *************************************************************/
function rmi_xlateServerCookie(cookieStr)
{
    var list = cookieStr.split(";");
    var ret = "";
    var hasDomain = false;
    var hasPath = false;

    for (var i = 0; i < list.length; ++i)
    {
        // NOTE: array's length > 2 if there are more than 2 '='

        var array = list[i].split("=");

        if (array.length < 1) continue;

        var key = rmi_trim(array[0]).toLowerCase();

        if (key == "domain")
        {
            hasDomain = true;
```

```
            var domainVal = rmi_trim(array[1]).toLowerCase();
            if ( ! rmi_verifyCookieDomain(domainVal) )
                return "";                                  // BAD domain - RETURN
""
        }

        if (key == "path") hasPath = true;

        if (i != 0) ret += ";"
        ret += array.join("=")
    }

    if (! hasDomain) ret += "; domain=" + rmi_Hostname;
    if (! hasPath) ret += "; path=" + rmi_Pathname;

    return ret;
}

// -->
```

Appendix B1

```
/*
 * Lexer.
 * Copyright (c) 1998-1999 New Generation Software (NGS) Oy
 *
 * Author: Markku Rossi <mtr@ngs.fi>
 */

/*
 * This library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Library General Public
 * License as published by the Free Software Foundation; either
 * version 2 of the License, or (at your option) any later version.
 *
 * This library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 * Library General Public License for more details.
 *
 * You should have received a copy of the GNU Library General Public
 * License along with this library; if not, write to the Free
 * Software Foundation, Inc., 59 Temple Place - Suite 330, Boston,
 * MA 02111-1307, USA.
 */

/*
 * The GNU Library General Public License may also be downloaded at
 * http://www.gnu.org/copyleft/gpl.html.
 */

/************************************************************
 *
 * This software was modified by Yahoo! Inc. under the terms
 * of the GNU Library General Public License(LGPL). For all
 * legal, copyright, and technical issues relating to how
 * this software can be used under GNU LGPL, please write to:
 *
 * GNU Compliance, Legal Dept., Yahoo! Inc.,
 * 3420 Central Expressway, Santa Clara, California U.S.A.
 *
 ************************************************************/

var rjs_VTAB = '\013';      // @@ For IE


/*
 * $Source: /usr/local/cvsroot/ngs/js/jsc/lexer.js,v $
 * $Id: lexer.js,v 1.9 1999/01/11 08:56:30 mtr Exp $
 */

/*
 * Global functions.
 */

function JSC$lexer (stream)
{
```

```
var ch, ch2;

JSC$token_value = null;

while ((ch = stream.readByte ()) != -1)
  {
    if (rjs_Error) return false;          //@@ avoid infinite loop

    if (ch == '\n')
      {
        JSC$linenum++;
        continue;
      }

    if (JSC$lexer_is_white_space (ch))
      continue;

    JSC$token_linenum = JSC$linenum;

    if (ch == '/' && JSC$lexer_peek_char (stream) == '*')
      {
        /* Multi line comment. */
        stream.readByte ();
        while ((ch = stream.readByte ()) != -1
               && (ch != '*' || JSC$lexer_peek_char (stream) != '/'))
          if (ch == '\n')
            JSC$linenum++;

        /* Consume the peeked '/' character. */
        stream.readByte ();
      }
    else if ((ch == '/' && JSC$lexer_peek_char (stream) == '/')
             || (ch == '#' && JSC$lexer_peek_char (stream) == '!'))
      {
        /* Single line comment. */
        while ((ch = stream.readByte ()) != -1 && ch != '\n')
          ;
        if (ch == '\n')
          JSC$linenum++;
      }
    else if (ch == '"' || ch == '\'')
      {
        /* String constant. */
        JSC$token_value = JSC$lexer_read_string (stream, "string", ch);
        return JSC$tSTRING;
      }

    /* Literals. */
    else if (ch == '=' && JSC$lexer_peek_char (stream) == '=')
      {
        stream.readByte ();
        if (JSC$lexer_peek_char (stream) == '=')
          {
            stream.readByte ();
            return JSC$tSEQUAL;
          }
        return JSC$tEQUAL;
```

```
  }
else if (ch == '!' && JSC$lexer_peek_char (stream) == '=')
  {
    stream.readByte ();
    if (JSC$lexer_peek_char (stream) == '=')
      {
        stream.readByte ();
        return JSC$tSNEQUAL;
      }
    return JSC$tNEQUAL;
  }
else if (ch == '<' && JSC$lexer_peek_char (stream) == '=')
  {
    stream.readByte ();
    return JSC$tLE;
  }
else if (ch == '>' && JSC$lexer_peek_char (stream) == '=')
  {
    stream.readByte ();
    return JSC$tGE;
  }
else if (ch == '&' && JSC$lexer_peek_char (stream) == '&')
  {
    stream.readByte ();
    return JSC$tAND;
  }
else if (ch == '|' && JSC$lexer_peek_char (stream) == '|')
  {
    stream.readByte ();
    return JSC$tOR;
  }
else if (ch == '+' && JSC$lexer_peek_char (stream) == '+')
  {
    stream.readByte ();
    return JSC$tPLUSPLUS;
  }
else if (ch == '-' && JSC$lexer_peek_char (stream) == '-')
  {
    stream.readByte ();
    return JSC$tMINUSMINUS;
  }
else if (ch == '*' && JSC$lexer_peek_char (stream) == '=')
  {
    stream.readByte ();
    return JSC$tMULA;
  }
else if (ch == '/' && JSC$lexer_peek_char (stream) == '=')
  {
    stream.readByte ();
    return JSC$tDIVA;
  }
else if (ch == '%' && JSC$lexer_peek_char (stream) == '=')
  {
    stream.readByte ();
    return JSC$tMODA;
  }
else if (ch == '+' && JSC$lexer_peek_char (stream) == '=')
```

```
    {
      stream.readByte ();
      return JSC$tADDA;
    }
  else if (ch == '-' && JSC$lexer_peek_char (stream) == '=')
    {
      stream.readByte ();
      return JSC$tSUBA;
    }
  else if (ch == '&' && JSC$lexer_peek_char (stream) == '=')
    {
      stream.readByte ();
      return JSC$tANDA;
    }
  else if (ch == '^' && JSC$lexer_peek_char (stream) == '=')
    {
      stream.readByte ();
      return JSC$tXORA;
    }
  else if (ch == '|' && JSC$lexer_peek_char (stream) == '=')
    {
      stream.readByte ();
      return JSC$tORA;
    }
  else if (ch == '<' && JSC$lexer_peek_char (stream) == '<')
    {
      stream.readByte ();
      if (JSC$lexer_peek_char (stream) == '=')
        {
          stream.readByte ();
          return JSC$tLSIA;
        }
      else
        return JSC$tLSHIFT;
    }
  else if (ch == '>' && JSC$lexer_peek_char (stream) == '>')
    {
      stream.readByte ();
      ch2 = JSC$lexer_peek_char (stream);
      if (ch2 == '=')
        {
          stream.readByte ();
          return JSC$tRSIA;
        }
      else if (ch2 == '>')
        {
          stream.readByte ();
          if (JSC$lexer_peek_char (stream) == '=')
            {
              stream.readByte ();
              return JSC$tRRSA;
            }
          else
            return JSC$tRRSHIFT;
        }
      else
        return JSC$tRSHIFT;
```

```
    }

  /* Identifiers and keywords. */
  else if (JSC$lexer_is_identifier_letter (ch))
    {
      /* An identifier. */
      //@@ var id = String.fromCharCode (ch);
      var id = "" + ch;

      while ((ch = stream.readByte ()) != -1
             && (JSC$lexer_is_identifier_letter (ch)
                 || JSC$lexer_is_decimal_digit (ch)))
        id += ch;       //@@ id.append (File.byteToString (ch));

      stream.ungetByte (ch);

      /* Keywords. */
      if (id == "break")
        return JSC$tBREAK;
      else if (id == "continue")
        return JSC$tCONTINUE;
      else if (id == "delete")
        return JSC$tDELETE;
      else if (id == "else")
        return JSC$tELSE;
      else if (id == "for")
        return JSC$tFOR;
      else if (id == "function")
        return JSC$tFUNCTION;
      else if (id == "if")
        return JSC$tIF;
      else if (id == "in")
        return JSC$tIN;
      else if (id == "new")
        return JSC$tNEW;
      else if (id == "return")
        return JSC$tRETURN;
      else if (id == "this")
        return JSC$tTHIS;
      else if (id == "typeof")
        return JSC$tTYPEOF;
      else if (id == "var")
        return JSC$tVAR;
      else if (id == "void")
        return JSC$tVOID;
      else if (id == "while")
        return JSC$tWHILE;
      else if (id == "with")
        return JSC$tWITH;

      /*
       * Future reserved keywords (some of these is already in use
       * in this implementation).
       */
      else if (id == "case")
        return JSC$tCASE;
      else if (id == "catch")
```

```
      return JSC$tCATCH;
    else if (id == "class")
      return JSC$tCLASS;
    else if (id == "const")
      return JSC$tCONST;
    else if (id == "debugger")
      return JSC$tDEBUGGER;
    else if (id == "default")
      return JSC$tDEFAULT;
    else if (id == "do")
      return JSC$tDO;
    else if (id == "enum")
      return JSC$tENUM;
    else if (id == "export")
      return JSC$tEXPORT;
    else if (id == "extends")
      return JSC$tEXTENDS;
    else if (id == "finally")
      return JSC$tFINALLY;
    else if (id == "import")
      return JSC$tIMPORT;
    else if (id == "super")
      return JSC$tSUPER;
    else if (id == "switch")
      return JSC$tSWITCH;
    else if (id == "throw")
      return JSC$tTHROW;
    else if (id == "try")
      return JSC$tTRY;

    /* Null and boolean literals. */
    else if (id == "null")
      return JSC$tNULL;
    else if (id == "true")
      return JSC$tTRUE;
    else if (id == "false")
      return JSC$tFALSE;
    else
      {
        /* It really is an identifier. */
        JSC$token_value = id;
        return JSC$tIDENTIFIER;
      }
  }

/* Character constants. */
else if (ch == '#' && JSC$lexer_peek_char (stream) == '\'')
  {
    /* Skip the starting '\'' and read more. */
    stream.readByte ();

    ch = stream.readByte ();
    if (ch == '\\')
      {
        JSC$token_value
          = JSC$lexer_read_backslash_escape (stream, 0, "character");
```

```
        if (stream.readByte () != '\'')
          error (JSC$filename + ":" + JSC$linenum.toString ()
                 + ": malformed character constant");
      }
    else if (JSC$lexer_peek_char (stream) == '\'')
      {
        stream.readByte ();
        JSC$token_value = ch;
      }
    else
      error (JSC$filename + ":" + JSC$linenum.toString ()
             + ": malformed character constant");

    return JSC$tINTEGER;
  }

/* Octal and hex numbers. */
else if (ch == '0'
         && JSC$lexer_peek_char (stream) != '.'
         && JSC$lexer_peek_char (stream) != 'e'
         && JSC$lexer_peek_char (stream) != 'E')
  {
    JSC$token_value = 0;
    ch = stream.readByte ();
    if (ch == 'x' || ch == 'X')
      {
        ch = stream.readByte ();
        while (JSC$lexer_is_hex_digit (ch))
          {
            JSC$token_value *= 16;
            JSC$token_value += JSC$lexer_hex_to_dec (ch);
            ch = stream.readByte ();
          }
        stream.ungetByte (ch);
      }
    else
      {
        while (JSC$lexer_is_octal_digit (ch))
          {
            JSC$token_value *= 8;
            JSC$token_value += ch - '0';
            ch = stream.readByte ();
          }
        stream.ungetByte (ch);
      }

    return JSC$tINTEGER;
  }

/* Decimal numbers. */
else if (JSC$lexer_is_decimal_digit (ch)
         || (ch == '.'
             && JSC$lexer_is_decimal_digit (
                                  JSC$lexer_peek_char (stream))))
  {
    var is_float = false;
    var buf = new String (File.byteToString (ch));
```

```
var accept_dot = true;

if (ch == '.')
  {
    /*
     * We started with '.' and we know that the next character
     * is a decimal digit (we peeked it).
     */
    is_float = true;

    ch = stream.readByte ();
    while (JSC$lexer_is_decimal_digit (ch))
      {
        buf += ch;    //@@ buf.append (File.byteToString (ch));
        ch = stream.readByte ();
      }
    accept_dot = false;
  }
else
  {
    /* We did start with a decimal digit. */
    ch = stream.readByte ();
    while (JSC$lexer_is_decimal_digit (ch))
      {
        buf += ch;    //@@ buf.append (File.byteToString (ch));
        ch = stream.readByte ();
      }
  }

if ((accept_dot && ch == '.')
    || ch == 'e' || ch == 'E')
  {
    is_float = true;

    if (ch == '.')
      {
        buf += ch;    //@@ buf.append (File.byteToString (ch));

        ch = stream.readByte ();
        while (JSC$lexer_is_decimal_digit (ch))
          {
            buf += ch;    //@@ buf.append (File.byteToString (ch));
            ch = stream.readByte ();
          }
      }

    if (ch == 'e' || ch == 'E')
      {
        buf += ch;    //@@ buf.append (File.byteToString (ch));
        ch = stream.readByte ();
        if (ch == '+' || ch == '-')
          {
            buf += ch;    //@@ buf.append (File.byteToString (ch));
            ch = stream.readByte ();
          }
        if (!JSC$lexer_is_decimal_digit (ch))
          error (JSC$filename + ":" + JSC$linenum.toString ()
```

```
                                + ": malformed exponent part in a decimal literal");

                        while (JSC$lexer_is_decimal_digit (ch))
                          {
                            buf += ch;    //@@ buf.append (File.byteToString (ch));
                            ch = stream.readByte ();
                          }
                      }
                  }

              /* Finally, we put the last character pack to the stream. */
              stream.ungetByte (ch);

              if (is_float)
                {
                  JSC$token_value = parseFloat (buf);
                  return JSC$tFLOAT;
                }

              JSC$token_value = parseInt (buf);
              return JSC$tINTEGER;
            }

          /* Just return the character as-is. */
          else
            return ch;
        }

      /* EOF reached. */
      return JSC$tEOF;
    }


    /*
     * Help functions.
     */

    function JSC$lexer_peek_char (stream)
    {
      var ch2 = stream.readByte ();
      stream.ungetByte (ch2);

      return ch2;
    }


    function JSC$lexer_is_identifier_letter (ch)
    {
      return (('a' <= ch && ch <= 'z') || ('A' <= ch && ch <= 'Z')
              || ch == '$' || ch == '_');
    }


    function JSC$lexer_is_octal_digit (ch)
    {
      return ('0' <= ch && ch <= '7');
    }
```

```
function JSC$lexer_is_decimal_digit (ch)
{
  return '0' <= ch && ch <= '9';
}


function JSC$lexer_is_hex_digit (ch)
{
  return (('0' <= ch && ch <= '9')
          || ('a' <= ch && ch <= 'f')
          || ('A' <= ch && ch <= 'F'));
}


function JSC$lexer_is_white_space (ch)
{
  //@@ return (ch == ' ' || ch == '\t' || ch == '\v' || ch == '\r'

  return (ch == ' ' || ch == '\t' || ch == rjs_VTAB || ch == '\r'
          || ch == '\f' || ch == '\n');
}


function JSC$lexer_hex_to_dec (ch)
{
  return (('0' <= ch && ch <= '9')
          ? ch - '0'
          : (('a' <= ch && ch <= 'f')
             ? 10 + ch - 'a'
             : 10 + ch - 'A'));
}


function JSC$lexer_read_backslash_escape (stream, possible_start, name)
{
  var ch = stream.readByte ();

  if (ch == 'n')
    ch = '\n';
  else if (ch == 't')
    ch = '\t';
  else if (ch == 'v')
    ch = rjs_VTAB;          //@@ ch = '\v';
  else if (ch == 'b')
    ch = '\b';
  else if (ch == 'r')
    ch = '\r';
  else if (ch == 'f')
    ch = '\f';
  else if (ch == 'a')
    ch = '\a';
  else if (ch == '\\')
    ch = '\\';
  else if (ch == '?')
    ch = '?';
```

```
else if (ch == '\'')
  ch = '\'';
else if (ch == '"')
  ch = '"';
else if (ch == 'x')
  {
    /* HexEscapeSequence. */
    var c1, c2;

    c1 = stream.readByte ();
    c2 = stream.readByte ();

    if (c1 == -1 || c2 == -1)
      JSC$lexer_eof_in_constant (possible_start, name);

    if (!JSC$lexer_is_hex_digit (c1) || !JSC$lexer_is_hex_digit (c2))
      error (JSC$filename + ":" + JSC$linenum.toString ()
             + ": \\x used with no following hex digits");

    ch = (JSC$lexer_hex_to_dec (c1) << 4) + JSC$lexer_hex_to_dec (c2);
  }
else if (ch == 'u')
  {
    /* UnicodeEscapeSequence. */
    var c1, c2, c3, c4;

    c1 = stream.readByte ();
    c2 = stream.readByte ();
    c3 = stream.readByte ();
    c4 = stream.readByte ();

    if (c1 == -1 || c2 == -1 || c3 == -1 || c4 == -1)
      JSC$lexer_eof_in_constant (possible_start, name);

    if (!JSC$lexer_is_hex_digit (c1) || !JSC$lexer_is_hex_digit (c2)
        || !JSC$lexer_is_hex_digit (c3) || !JSC$lexer_is_hex_digit (c4))
      error (JSC$filename + ":" + JSC$linenum.toString ()
             + ": \\u used with no following hex digits");

    ch = ((JSC$lexer_hex_to_dec (c1) << 12)
          + (JSC$lexer_hex_to_dec (c2) << 8)
          + (JSC$lexer_hex_to_dec (c3) << 4)
          + JSC$lexer_hex_to_dec (c4));
  }
else if (JSC$lexer_is_octal_digit (ch))
  {
    var result = ch - '0';
    var i = 1;

    if (ch == '0')
      /* Allow three octal digits after '0'. */
      i = 0;

    ch = stream.readByte ();
    while (i < 3 && JSC$lexer_is_octal_digit (ch))
      {
        result *= 8;
```

```
            result += ch - '0';
            ch = stream.readByte ();
            i++;
          }
        stream.ungetByte (ch);
        ch = result;
      }
    else
      {
        if (ch == -1)
          error (JSC$filename + ":" + JSC$linenum.toString ()
                 + ": unterminated " + name);

        JSC$warning (JSC$filename + ":" + JSC$linenum.toString ()
                     + ": warning: unknown escape sequence `\\"
                     + File.byteToString (ch) + "'");
      }

    return ch;
  }


  function JSC$lexer_read_string (stream, name, ender)
  {
    var str = new String ("");
    var done = false, ch;
    var possible_start_ln = JSC$linenum;
    var warned_line_terminator = false;

    while (!done)
      {
        if (rjs_Error) return false;          //@@ avoid infinite loop

        ch = stream.readByte ();
        if (ch == '\n')
          {
            if (JSC$warn_strict_ecma && !warned_line_terminator)
              {
                JSC$warning (JSC$filename + ":" + JSC$linenum.toString ()
                             + ": warning: ECMAScript don't allow line terminators
in "
                             + name + " constants");
                warned_line_terminator = true;
              }
            JSC$linenum++;
          }

        if (ch == -1)
          JSC$lexer_eof_in_constant (possible_start_ln, name);

        else if (ch == ender)
          done = true;
        else
          {
            if (ch == '\\')
              {
                if (JSC$lexer_peek_char (stream) == '\n')
```

```
                {
                  /*
                   * Backslash followed by a newline character.  Ignore
                   * them both.
                   */
                  stream.readByte ();
                  JSC$linenum++;
                  continue;
                }
              ch = JSC$lexer_read_backslash_escape (stream, possible_start_ln,
                                                    name);
            }
          str += ch;    //@@ str.append (ch);
        }
    }

  return str;
}


function JSC$lexer_read_regexp_constant (stream)
{
  /* Regexp literal. */
  var source = JSC$lexer_read_regexp_source (stream);

  /* Check the possible flags. */
  var flags = new String ("");
  while ((ch = JSC$lexer_peek_char (stream)) == 'g' || ch == 'i')
    {
      stream.readByte ();
      flags += ch;    //@@ flags.append (File.byteToString (ch));
    }

  /* Try to compile it. */
  var msg = false;
  var result;

  //@@
  result = new RegExp (source, flags);

  /*** @@
  try
    {
      result = new RegExp (source, flags);
    }
  catch (msg)
    {
      var start = msg.lastIndexOf (":");
      msg = (JSC$filename + ":" + JSC$token_linenum.toString ()
             + ": malformed regular expression constant:"
             + msg.substr (start + 1));
    }
  ***/

  if (msg)
    error (msg);
```

```
  /* Success. */

  return result;
}


function JSC$lexer_read_regexp_source (stream)
{
  var str = new String ("");
  var done = false, ch;
  var possible_start_ln = JSC$linenum;
  var warned_line_terminator = false;
  var name = "regular expression";

  while (!done)
    {
      if (rjs_Error) return false;          //@@ avoid infinite loop

      ch = stream.readByte ();

      if (ch == '\n')
        {
          if (JSC$warn_strict_ecma && !warned_line_terminator)
            {
              JSC$warning (JSC$filename + ":" + JSC$linenum.toString ()
                           + ": warning: ECMAScript don't allow line "
                           + "terminators in " + name + " constants");
              warned_line_terminator = true;
            }
          JSC$linenum++;
        }

      if (ch == -1)
        JSC$lexer_eof_in_constant (possible_start_ln, name);

      else if (ch == '/')
        done = true;
      else
        {
          if (ch == '\\')
            {
              ch = stream.readByte ();
              if (ch == '\n')
                {
                  /*
                   * Backslash followed by a newline character.  Ignore
                   * them both.
                   */
                  JSC$linenum++;
                  continue;
                }
              if (ch == -1)
                JSC$lexer_eof_in_constant (possible_start_ln, name);

              /* Handle the backslash escapes. */
              if (ch == 'f')
                ch = '\f';
```

```
                else if (ch == 'n')
                  ch = '\n';
                else if (ch == 'r')
                  ch = '\r';
                else if (ch == 't')
                  ch = '\t';
                else if (ch == 'v')
                  ch = rjs_VTAB;          //@@ Bug with '==' from original codes?
ch == '\v';
                else if (ch == 'c')
                  {
                    /* SourceCharacter. */
                    ch = stream.readByte ();
                    if (ch == -1)
                      JSC$lexer_eof_in_constant (possible_start_ln, name);

                    if (ch == '\n' && JSC$warn_strict_ecma)
                      JSC$warning (JSC$filename + ":" + JSC$linenum.toString ()
                                   + ": warning: ECMAScript don't allow line
termiantor after \\c in regular expression constants");

                    /*
                     * Append the source-character escape start.  The ch
                     * will be appended later.
                     */
                    str += "\\c";    //@@ str.append ("\\c");
                  }
                else if (ch == 'u' || ch == 'x' || ch == '0')
                  {
                    /* These can be handled with the read_backslash_escape(). */
                    stream.ungetByte (ch);
                    ch = JSC$lexer_read_backslash_escape (stream);
                  }
                else
                  {
                    /*
                     * Nothing special.  Leave it to the result as-is.
                     * The regular expression backage will handle it.
                     */
                    stream.ungetByte (ch);
                    ch = '\\';
                  }
              }
            str += ch;    //@@ str.append (File.byteToString (ch));
          }
      }

  return str;
}


function JSC$lexer_eof_in_constant (possible_start, name)
{
  var msg = (JSC$filename + ":" + JSC$linenum.toString ()
             + ": unterminated " + name + " constant");

  if (possible_start > 0)
```

```
    {
      //@@ msg += (System.lineBreakSequence

      msg += ("\n"
              + JSC$filename + ":" + possible_start.toString ()
              + ": possible real start of unterminated " + name + " constant");
    }

    error (msg);
  }
```

```
/*
Local variables:
mode: c
End:
*/
```

```
/*
 * Parser.
 * Copyright (c) 1998 New Generation Software (NGS) Oy
 *
 * Author: Markku Rossi <mtr@ngs.fi>
 */

/*
 * This library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Library General Public
 * License as published by the Free Software Foundation; either
 * version 2 of the License, or (at your option) any later version.
 *
 * This library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 * Library General Public License for more details.
 *
 * You should have received a copy of the GNU Library General Public
 * License along with this library; if not, write to the Free
 * Software Foundation, Inc., 59 Temple Place - Suite 330, Boston,
 * MA 02111-1307, USA
 */

/*
 * The GNU Library General Public License may also be downloaded at
 * http://www.gnu.org/copyleft/gpl.html.
 */

/************************************************************
 *
 * This software was modified by Yahoo! Inc. under the terms
 * of the GNU Library General Public License(LGPL). For all
 * legal, copyright, and technical issues relating to how
 * this software can be used under GNU LGPL, please write to:
 *
 * GNU Compliance, Legal Dept., Yahoo! Inc.,
 * 3420 Central Expressway, Santa Clara, California U.S.A.
 *
 ************************************************************/


 /*
 * $Source: /usr/local/cvsroot/ngs/js/jsc/parser.js,v $
 * $Id: parser.js,v 1.26 1998/10/26 15:25:21 mtr Exp $
 */

/*
 * Global functions.
 */

function JSC$parser_reset ()
{
  JSC$function = null;
  JSC$global_stmts = null;
  JSC$nested_function_declarations = null;
}
```

```
function JSC$parser_parse (stream)
{
  JSC$linenum = 1;
  JSC$filename = stream.name;
  JSC$functions = new Array ();
  JSC$global_stmts = new Array ();
  JSC$nested_function_declarations = new Array ();
  JSC$anonymous_function_count = 0;
  JSC$parser_peek_token_valid = false;
  JSC$num_tokens = 0;
  JSC$num_arguments_identifiers = 0;
  JSC$num_missing_semicolons = 0;

  if (JSC$verbose)
    JSC$message ("jsc: parsing");

  while (JSC$parser_peek_token (stream) != JSC$tEOF)
    if (!JSC$parser_parse_source_element (stream))
    {
      JSC$parser_syntax_error ();
      return false;                    //@@ avoid infinite loop
    }


  if (JSC$verbose)
    {
      var msg = ("jsc: input stream had " + (JSC$linenum - 1).toString ()
                 + " lines, " + JSC$num_tokens.toString () + " tokens");

      if (JSC$num_missing_semicolons > 0)
        msg += (", " + JSC$num_missing_semicolons.toString ()
                + " missing semicolons");

      JSC$message (msg);
    }
}


/*
 * General help functions.
 */

function JSC$parser_syntax_error ()
{
  error (JSC$filename + ":" + JSC$linenum.toString () + ": syntax error");
}

/* All warnings are reported through this function. */
function JSC$warning (line)
{
    rjs_warn(line);    //@@ System.stderr.writeln (line);
}

/* All messages are reported throught this function. */
function JSC$message (line)
```

```
{
    rjs_info(line);     //@@ System.stderr.writeln (line);
}


function JSC$parser_get_token (stream)
{
  JSC$num_tokens++;

  var token;
  if (JSC$parser_peek_token_valid)
    {
      JSC$parser_peek_token_valid = false;
      JSC$parser_token_value = JSC$parser_peek_token_value;
      JSC$parser_token_linenum = JSC$parser_peek_token_linenum;
      token = JSC$parser_peek_token_token;
    }
  else
    {
      token = JSC$lexer (stream);
      JSC$parser_token_value = JSC$token_value;
      JSC$parser_token_linenum = JSC$token_linenum;
    }

  if (token == JSC$tIDENTIFIER && JSC$parser_token_value == "arguments")
    JSC$num_arguments_identifiers++;

  return token;
}


function JSC$parser_peek_token (stream)
{
  if (JSC$parser_peek_token_valid)
    return JSC$parser_peek_token_token;
  else
    {
      JSC$parser_peek_token_token = JSC$lexer (stream);
      JSC$parser_peek_token_value = JSC$token_value;
      JSC$parser_peek_token_linenum = JSC$token_linenum;
      JSC$parser_peek_token_valid = true;
      return JSC$parser_peek_token_token;
    }
}


function JSC$parser_get_semicolon_asci (stream)
{
  var token = JSC$parser_peek_token (stream);

  if (token == ';')
    {
      rjs_Tokens.push(";");     //@@

      /* Everything ok.  It was there. */
      return JSC$parser_get_token (stream);
    }
```

```
  /* No semicolon.  Let's see if we can insert it there. */
  if (token == '}'
      || JSC$parser_token_linenum < JSC$parser_peek_token_linenum
      || token == JSC$tEOF)
    {
      rjs_Tokens.push(";");     //@@

      /* Ok, do the automatic semicolon insertion. */
      if (JSC$warn_missing_semicolon)
        JSC$warning (JSC$filename + ":" + JSC$parser_token_linenum.toString ()
                     + ": warning: missing semicolon");
      JSC$num_missing_semicolons++;
      return ';';
    }

  /* Sorry, no can do. */
  JSC$parser_syntax_error ();
}


function JSC$parser_expr_is_left_hand_side (expr)
{
  return (expr.etype == JSC$EXPR_CALL
          || expr.etype == JSC$EXPR_OBJECT_PROPERTY
          || expr.etype == JSC$EXPR_OBJECT_ARRAY
          || expr.etype == JSC$EXPR_NEW
          || expr.etype == JSC$EXPR_THIS
          || expr.etype == JSC$EXPR_IDENTIFIER
          || expr.etype == JSC$EXPR_FLOAT
          || expr.etype == JSC$EXPR_INTEGER
          || expr.etype == JSC$EXPR_STRING
          || expr.etype == JSC$EXPR_REGEXP
          || expr.etype == JSC$EXPR_ARRAY_INITIALIZER
          || expr.etype == JSC$EXPR_NULL
          || expr.etype == JSC$EXPR_TRUE
          || expr.etype == JSC$EXPR_FALSE);
}


function JSC$parser_parse_source_element (stream)
{
  rjs_Tokens.reset();     //@@

  if (JSC$parser_parse_function_declaration (stream))
  {
    rjs_Stmts.push( rjs_Tokens.str() );    //@@ save one statement
    return true;
  }

  rjs_Tokens.reset();     //@@

  var stmt = JSC$parser_parse_stmt (stream);

  if (!stmt)
    return false;
```

```
  if (stmt.stype == JSC$STMT_VARIABLE)
    /*
     * This is a variable declaration at the global level.  These
     * are actually global variables.
     */
    stmt.global_level = true;

  rjs_xDomain();                          //@@
  rjs_xLocation();                        //@@
  rjs_xCookie();                          //@@
  rjs_Stmts.push( rjs_Tokens.str() );     //@@ save one statement

  JSC$global_stmts.push (stmt);

  return true;
}


function JSC$parser_parse_function_declaration (stream)
{
  var id, args, block;

  if (JSC$parser_peek_token (stream) != JSC$tFUNCTION)
    return false;

  rjs_Tokens.push("function ");  //@@

  /* Record how many `arguments' identifiers have been seen so far. */
  var num_arguments_identifiers = JSC$num_arguments_identifiers;

  JSC$parser_get_token (stream);
  if (JSC$parser_get_token (stream) != JSC$tIDENTIFIER)
    JSC$parser_syntax_error ();

  id = JSC$parser_token_value;
  var ln = JSC$parser_token_linenum;
  var id_given = id;

  rjs_Tokens.push(id_given);    //@@

  if (JSC$nested_function_declarations.length > 0)
    {
      /* This is a nested function declaration. */
      id = ".F:" + (JSC$anonymous_function_count++).toString ();
    }
  JSC$nested_function_declarations.push (id);

  if (JSC$parser_get_token (stream) != '(')
    JSC$parser_syntax_error ();

  rjs_Tokens.push("(");    //@@

  /* Formal parameter list opt. */
  args = new Array ();
  while (JSC$parser_peek_token (stream) != ')')
    {
      if (rjs_Error) return false;          //@@ avoid infinite loop
```

```
      if (JSC$parser_get_token (stream) != JSC$tIDENTIFIER)
        JSC$parser_syntax_error ();
      args.push (JSC$parser_token_value);

      rjs_Tokens.push(JSC$parser_token_value);    //@@

      var token = JSC$parser_peek_token (stream);
      if (token == ',')
        {
          rjs_Tokens.push(",");    //@@

          JSC$parser_get_token (stream);
          if (JSC$parser_peek_token (stream) != JSC$tIDENTIFIER)
            JSC$parser_syntax_error ();
        }
      else if (token != ')')
        JSC$parser_syntax_error ();
    }

  if (JSC$parser_get_token (stream) != ')')
    JSC$parser_syntax_error ();

  rjs_Tokens.push(") ");    //@@

  JSC$parser_peek_token (stream);
  var lbrace_ln = JSC$parser_peek_token_linenum;

  block = JSC$parser_parse_block (stream);
  if (typeof block == "boolean")
    JSC$parser_syntax_error ();

  /* Did the function use the `arguments' identifier? */
  var use_arguments = false;
  if (JSC$num_arguments_identifiers > num_arguments_identifiers)
    {
      use_arguments = true;
      if (JSC$warn_deprecated)
        JSC$warning (JSC$filename + ":" + ln.toString ()
                     + ": warning: the `arguments' property of Function "
                     + "instance is deprecated");
    }

  JSC$functions.push (new JSC$function_declaration (ln, lbrace_ln, id,
                                                    id_given, args,
                                                    block, use_arguments));

  JSC$nested_function_declarations.pop ();

  return true;
}


function JSC$parser_parse_block (stream)
{
  var block;
```

```
  if (JSC$parser_peek_token (stream) != '{')
    return false;

  //@@ original NGS bug ?? JSC$parser_get_token (stream) != '{';
  JSC$parser_get_token (stream);

  rjs_Tokens.push("{");  //@@

  var ln = JSC$parser_peek_token_linenum;

  /* Do we have a statement list? */
  if (JSC$parser_peek_token (stream) != '}')
    /* Yes we have. */
    block = JSC$parser_parse_stmt_list (stream);
  else
    /* Do we don't */
    block = new Array ();

  if (JSC$parser_get_token (stream) != '}')
    JSC$parser_syntax_error ();

  rjs_Tokens.push("}");  //@@

  block.linenum = ln;

  return block;
}


function JSC$parser_parse_stmt_list (stream)
{
  var list, done, item;

  list = new Array ();
  done = false;

  while (!done)
    {
      if (rjs_Error) return false;          //@@ avoid infinite loop

      item = JSC$parser_parse_stmt (stream);
      if (typeof item == "boolean")
        {
          /* Can't parse more statements.  We'r done. */
          done = true;
        }
      else
        list.push (item);
    }

  return list;
}


function JSC$parser_parse_stmt (stream)
{
  var item, token;
```

```
if (typeof (item = JSC$parser_parse_block (stream)) != "boolean")
  return new JSC$stmt_block (item.linenum, item);
else if (JSC$parser_parse_function_declaration (stream))
  {
    //@@
    /* XXX The function declaration as statement might be incomplete. */

    if (JSC$nested_function_declarations.length == 0)
      /* Function declaration at top-level statements. */
      return new JSC$stmt_empty (JSC$parser_token_linenum);

    /* Function declaration inside another function. */

    var container_id = JSC$nested_function_declarations.pop ();
    JSC$nested_function_declarations.push (container_id);

    var f = JSC$functions[JSC$functions.length - 1];
    var function_id = f.name;
    var given_id = f.name_given;

    return new JSC$stmt_function_declaration (JSC$parser_token_linenum,
                                              container_id, function_id,
                                              given_id);
  }
else if (typeof (item = JSC$parser_parse_variable_stmt (stream))
         != "boolean")
  return item;
else if (typeof (item = JSC$parser_parse_if_stmt (stream))
         != "boolean")
  return item;
else if (typeof (item = JSC$parser_parse_iteration_stmt (stream))
         != "boolean")
  return item;
else if (typeof (item = JSC$parser_parse_expr (stream))
         != "boolean")
  {
    if (item.etype == JSC$EXPR_IDENTIFIER)
      {
        /* Possible `Labeled Statement'. */
        token = JSC$parser_peek_token (stream);
        if (token == ':' && item.linenum == JSC$parser_peek_token_linenum)
          {
            /* Yes it is. */
            JSC$parser_get_token (stream);

            rjs_Tokens.push(": ");        //@@

            var stmt = JSC$parser_parse_stmt (stream);
            if (!stmt)
              JSC$parser_syntax_error;

            return new JSC$stmt_labeled_stmt (item.linenum, item.value,
                                              stmt);
          }
        /* FALLTHROUGH */
      }
```

```
      JSC$parser_get_semicolon_asci (stream);

      return new JSC$stmt_expr (item);
    }
  else
    {
      token = JSC$parser_peek_token (stream);
      if (token == ';')
        {
          rjs_Tokens.push(";");     //@@

          JSC$parser_get_token (stream);
          return new JSC$stmt_empty (JSC$parser_token_linenum);
        }
      else if (token == JSC$tCONTINUE)
        {
          rjs_Tokens.push("continue ");     //@@

          JSC$parser_get_token (stream);

          /* Check the possible label. */
          var label = null;
          token = JSC$parser_peek_token (stream);
          if (token == JSC$tIDENTIFIER
              && JSC$parser_token_linenum == JSC$parser_peek_token_linenum)
            {
              JSC$parser_get_token (stream);
              label = JSC$parser_token_value;

              rjs_Tokens.push(label);     //@@
            }

          item = new JSC$stmt_continue (JSC$parser_token_linenum, label);

          JSC$parser_get_semicolon_asci (stream);

          return item;
        }
      else if (token == JSC$tBREAK)
        {
          JSC$parser_get_token (stream);

          rjs_Tokens.push("break ");     //@@

          /* Check the possible label. */
          var label = null;
          token = JSC$parser_peek_token (stream);
          if (token == JSC$tIDENTIFIER
              && JSC$parser_token_linenum == JSC$parser_peek_token_linenum)
            {
              JSC$parser_get_token (stream);
              label = JSC$parser_token_value;

              rjs_Tokens.push(label);     //@@
            }
```

```
      item = new JSC$stmt_break (JSC$parser_token_linenum, label);

      JSC$parser_get_semicolon_asci (stream);

      return item;
    }
  else if (token == JSC$tRETURN)
    {
      JSC$parser_get_token (stream);
      var linenum = JSC$parser_token_linenum;

      rjs_Tokens.push("return ");    //@@

      if (JSC$parser_peek_token (stream) == ';')
        {
          /* Consume the semicolon. */
          JSC$parser_get_token (stream);
          item = null;

          rjs_Tokens.push(";");    //@@
        }
      else
        {
          if (JSC$parser_peek_token_linenum > linenum)
            {
              /*
               * A line terminator between tRETURN and the next
               * token that is not a semicolon.  ASCI here.
               */
              if (JSC$warn_missing_semicolon)
                JSC$warning (JSC$filename + ":" + linenum.toString ()
                             + ": warning: missing semicolon");

              JSC$num_missing_semicolons++;
              item = null;
            }
          else
            {
              item = JSC$parser_parse_expr (stream);
              if (typeof item == "boolean")
                JSC$parser_syntax_error ();

              JSC$parser_get_semicolon_asci (stream);
            }
        }

      return new JSC$stmt_return (linenum, item);
    }
  else if (token == JSC$tSWITCH)  //@@
    {
      JSC$parser_get_token (stream);
      return JSC$parser_parse_switch (stream);
    }
  else if (token == JSC$tWITH)
    {
      rjs_Tokens.push("with ");    //@@
```

```
      JSC$parser_get_token (stream);
      var linenum = JSC$parser_token_linenum;

      if (JSC$parser_get_token (stream) != '(')
        JSC$parser_syntax_error ();

      rjs_Tokens.push("(");    //@@

      var expr = JSC$parser_parse_expr (stream);
      if (typeof expr == "boolean")
        JSC$parser_syntax_error ();

      if (JSC$parser_get_token (stream) != ')')
        JSC$parser_syntax_error ();

      rjs_Tokens.push(") ");    //@@

      var stmt = JSC$parser_parse_stmt (stream);
      if (typeof stmt == "boolean")
        JSC$parser_syntax_error ();

      return new JSC$stmt_with (linenum, expr, stmt);
    }
  else if (token == JSC$tTRY)  //@@
    {
      JSC$parser_get_token (stream);
      return JSC$parser_parse_try (stream);
    }
  else if (token == JSC$tTHROW)  //@@
    {
      JSC$parser_get_token (stream);
      var linenum = JSC$parser_token_linenum;

      /*
       * Get the next token's linenum.  We need it for strict_ecma
       * warning.
       */
      JSC$parser_peek_token (stream);
      var peek_linenum = JSC$parser_peek_token_linenum;

      /* The expression to throw. */
      var expr = JSC$parser_parse_expr (stream);
      if (typeof expr == "boolean")
        JSC$parser_syntax_error ();

      if (JSC$warn_strict_ecma && peek_linenum > linenum)
        JSC$warning (JSC$filename + ":" + JSC$linenum.toString ()
                     + ": warning: ECMAScript don't allow line terminators"
                     + " between `throw' and expression");

      JSC$parser_get_semicolon_asci (stream);

      return new JSC$stmt_throw (linenum, expr);
    }
  else
    /* Can't parse more.  We'r done. */
    return false;
```

```
    }
}


function JSC$parser_parse_switch (stream)
{
  var linenum = JSC$parser_token_linenum;

  if (JSC$parser_get_token (stream) != '(')
    JSC$parser_syntax_error ();

  var expr = JSC$parser_parse_expr (stream);
  if (!expr)
    JSC$parser_syntax_error ();

  if (JSC$parser_get_token (stream) != ')')
    JSC$parser_syntax_error ();

  if (JSC$parser_get_token (stream) != '{')
    JSC$parser_syntax_error ();

  /* Parse case clauses. */
  var clauses = new Array ();
  while (true)
    {
      if (rjs_Error) return false;          //@@ avoid infinite loop

      var token = JSC$parser_get_token (stream);

      if (token == '}')
        break;
      else if (token == JSC$tCASE || token == JSC$tDEFAULT)
        {
          var stmts = new Array ();
          stmts.expr = null;

          if (token == JSC$tCASE)
            {
              stmts.expr = JSC$parser_parse_expr (stream);
              if (!stmts.expr)
                JSC$parser_syntax_error ();
            }
          if (JSC$parser_get_token (stream) != ':')
            JSC$parser_syntax_error ();

          stmts.linenum = JSC$parser_token_linenum;

          /* Read the statement list. */
          while (true)
            {
              if (rjs_Error) return false;          //@@ avoid infinite loop

              token = JSC$parser_peek_token (stream);
              if (token == '}' || token == JSC$tCASE || token == JSC$tDEFAULT)
                /* Done with this branch. */
                break;
```

```
              var stmt = JSC$parser_parse_stmt (stream);
              if (!stmt)
                JSC$parser_syntax_error ();

              stmts.push (stmt);
            }

          stmts.last_linenum = JSC$parser_token_linenum;

          /* One clause parsed. */
          clauses.push (stmts);
        }
      else
        JSC$parser_syntax_error ();
    }

  return new JSC$stmt_switch (linenum, JSC$parser_token_linenum, expr,
                              clauses);
}


function JSC$parser_parse_try (stream)
{
  var linenum = JSC$parser_token_linenum;

  var block = JSC$parser_parse_stmt (stream);
  if (!block)
    JSC$parser_syntax_error ();

  var try_block_last_linenum = JSC$parser_token_linenum;

  /* Now we must see `catch' or `finally'. */
  var token = JSC$parser_peek_token (stream);
  if (token != JSC$tCATCH && token != JSC$tFINALLY)
    JSC$parser_syntax_error ();

  var catch_list = false;
  if (token == JSC$tCATCH)
    {
      /* Parse catch list. */

      catch_list = new Array ();
      catch_list.linenum = JSC$parser_peek_token_linenum;

      while (token == JSC$tCATCH)
        {
          if (rjs_Error) return false;          //@@ avoid infinite loop

          JSC$parser_get_token (stream);
          var c = new Object ();
          c.linenum = JSC$parser_token_linenum;

          if (JSC$parser_get_token (stream) != '(')
            JSC$parser_syntax_error ();

          if (JSC$parser_get_token (stream) != JSC$tIDENTIFIER)
            JSC$parser_syntax_error ();
```

```
            c.id = JSC$parser_token_value;

            c.guard = false;
            if (JSC$parser_peek_token (stream) == JSC$tIF)
              {
                JSC$parser_get_token (stream);
                c.guard = JSC$parser_parse_expr (stream);
                if (!c.guard)
                  JSC$parser_syntax_error ();
              }

            if (JSC$parser_get_token (stream) != ')')
              JSC$parser_syntax_error ();

            c.stmt = JSC$parser_parse_stmt (stream);
            if (!c.stmt)
              JSC$parser_syntax_error ();

            catch_list.push (c);

            token = JSC$parser_peek_token (stream);
          }

        catch_list.last_linenum = JSC$parser_token_linenum;
      }

    var fin = false;
    if (token == JSC$tFINALLY)
      {
        /* Parse the finally. */
        JSC$parser_get_token (stream);

        fin = JSC$parser_parse_stmt (stream);
        if (!fin)
          JSC$parser_syntax_error ();
      }

    return new JSC$stmt_try (linenum, try_block_last_linenum,
                             JSC$parser_token_linenum, block, catch_list,
                             fin);
}


function JSC$parser_parse_variable_stmt (stream)
{
  var list, id, expr, token;

  if (JSC$parser_peek_token (stream) != JSC$tVAR)
    return false;

  JSC$parser_get_token (stream);
  var ln = JSC$parser_token_linenum;

  rjs_Tokens.push("var ");  //@@

  list = new Array ();
```

```
  while (true)
    {
      if (rjs_Error) return false;          //@@ avoid infinite loop

      token = JSC$parser_peek_token (stream);
      if (token == JSC$tIDENTIFIER)
        {
          JSC$parser_get_token ();
          id = JSC$parser_token_value;

          rjs_Tokens.push(id);  //@@

          if (JSC$parser_peek_token (stream) == '=')
            {
              rjs_Tokens.push("=");  //@@

              JSC$parser_get_token (stream);
              expr = JSC$parser_parse_assignment_expr (stream);
              if (typeof expr == "boolean")
                JSC$parser_syntax_error ();
            }
          else
            expr = null;


          list.push (new JSC$var_declaration (id, expr));
          // @@ rjs_debug("JSC$parser_parse_variable_stmt: var " + id + " = " +
expr.value);

          /* Check if we have more input. */
          if (JSC$parser_peek_token (stream) == ',')
            {
              /* Yes we have.  */
              JSC$parser_get_token (stream);

              rjs_Tokens.push(",");  //@@

              /* The next token must be tIDENTIFIER. */
              if (JSC$parser_peek_token (stream) != JSC$tIDENTIFIER)
                JSC$parser_syntax_error ();
            }
          else
            {
              /* No, we don't. */
              JSC$parser_get_semicolon_asci (stream);
              break;
            }
        }
      else
        {
          /* We'r done. */
          JSC$parser_get_semicolon_asci (stream);
          break;
        }
    }

  /* There must be at least one variable declaration. */
```

```
  if (list.length == 0)
      JSC$parser_syntax_error ();

  return new JSC$stmt_variable (ln, list);
}


function JSC$parser_parse_if_stmt (stream)
{
  var expr, stmt, stmt2;

  if (JSC$parser_peek_token (stream) != JSC$tIF)
    return false;

  rjs_Tokens.push(" if ");     //@@

  JSC$parser_get_token (stream);
  var ln = JSC$parser_token_linenum;

  if (JSC$parser_get_token (stream) != '(')
    JSC$parser_syntax_error ();

  rjs_Tokens.push("(");     //@@

  expr = JSC$parser_parse_expr (stream);
  if (typeof expr == "boolean")
    JSC$parser_syntax_error ();

  if (JSC$parser_get_token (stream) != ')')
    JSC$parser_syntax_error ();

  rjs_Tokens.push(") ");     //@@

  stmt = JSC$parser_parse_stmt (stream);
  if (typeof stmt == "boolean")
    JSC$parser_syntax_error ();

  if (JSC$parser_peek_token (stream) == JSC$tELSE)
    {
      rjs_Tokens.push(" else ");     //@@

      JSC$parser_get_token (stream);
      stmt2 = JSC$parser_parse_stmt (stream);
      if (typeof stmt2 == "boolean")
        JSC$parser_syntax_error ();
    }
  else
    stmt2 = null;

  return new JSC$stmt_if (ln, expr, stmt, stmt2);
}


function JSC$parser_parse_iteration_stmt (stream)
{
  var token, expr1, expr2, expr3, stmt;
```

```
token = JSC$parser_peek_token (stream);
if (token == JSC$tDO)
  {
    rjs_Tokens.push(" do ");    //@@

    /* do Statement while (Expression); */
    JSC$parser_get_token (stream);
    var ln = JSC$parser_token_linenum;

    stmt = JSC$parser_parse_stmt (stream);
    if (typeof stmt == "boolean")
      JSC$parser_syntax_error ();

    if (JSC$parser_get_token (stream) != JSC$tWHILE)
      JSC$parser_syntax_error ();

    rjs_Tokens.push(" while ");    //@@

    if (JSC$parser_get_token (stream) != '(')
      JSC$parser_syntax_error ();

    rjs_Tokens.push("(");    //@@

    expr1 = JSC$parser_parse_expr (stream);
    if (typeof expr1 == "boolean")
      JSC$parser_syntax_error ();

    if (JSC$parser_get_token (stream) != ')')
      JSC$parser_syntax_error ();

    rjs_Tokens.push(")");    //@@

    JSC$parser_get_semicolon_asci (stream);

    return new JSC$stmt_do_while (ln, expr1, stmt);
  }
else if (token == JSC$tWHILE)
  {
    rjs_Tokens.push(" while ");    //@@

    /* while (Expression) Statement */
    JSC$parser_get_token (stream);
    var ln = JSC$parser_token_linenum;

    if (JSC$parser_get_token (stream) != '(')
      JSC$parser_syntax_error ();

    rjs_Tokens.push(" ( ");    //@@

    expr1 = JSC$parser_parse_expr (stream);
    if (typeof expr1 == "boolean")
      JSC$parser_syntax_error ();

    if (JSC$parser_get_token (stream) != ')')
      JSC$parser_syntax_error ();

    rjs_Tokens.push(" ) ");    //@@
```

```
      stmt = JSC$parser_parse_stmt (stream);
      if (typeof stmt == "boolean")
        JSC$parser_syntax_error ();

      return new JSC$stmt_while (ln, expr1, stmt);
    }
  else if (token == JSC$tFOR)
    {
      rjs_Tokens.push(" for ");     //@@

      JSC$parser_get_token (stream);
      var ln = JSC$parser_token_linenum;

      if (JSC$parser_get_token (stream) != '(')
        JSC$parser_syntax_error ();

      rjs_Tokens.push("(");     //@@

      /* Init */

      var vars = null;

      token = JSC$parser_peek_token (stream);
      if (token == JSC$tVAR)
        {
          JSC$parser_get_token (stream);

          rjs_Tokens.push("var ");     //@@

          vars = new Array ();

          while (true)
            {
              if (rjs_Error) return false;          //@@ avoid infinite loop

              /* The identifier. */
              token = JSC$parser_peek_token (stream);
              if (token != JSC$tIDENTIFIER)
                break;

              JSC$parser_get_token (stream);
              var id = JSC$parser_token_value;

              rjs_Tokens.push(id);     //@@

              /* Possible initializer. */
              var expr = null;
              if (JSC$parser_peek_token (stream) == '=')
                {
                  JSC$parser_get_token (stream);

                  rjs_Tokens.push("=");     //@@

                  expr = JSC$parser_parse_assignment_expr (stream);
                  if (!expr)
                    JSC$parser_syntax_error ();
```

```
            }

          vars.push (new JSC$var_declaration (id, expr));

          /* Check if we have more input. */
          if (JSC$parser_peek_token (stream) == ',')
            {
              /* Yes we have. */
              JSC$parser_get_token (stream);

              rjs_Tokens.push(",");     //@@

              /* The next token must be tIDENTIFIER. */
              if (JSC$parser_peek_token (stream) != JSC$tIDENTIFIER)
                JSC$parser_syntax_error ();
            }
          else
            /* No more input. */
            break;
        }


      /* Must have at least one variable declaration. */
      if (vars.length == 0)
        JSC$parser_syntax_error ();
    }
  else if (token != ';')
    {
      expr1 = JSC$parser_parse_expr (stream);
      if (typeof expr1 == "boolean")
        JSC$parser_syntax_error ();
    }
  else
    expr1 = null;

  token = JSC$parser_get_token (stream);
  var for_in = false;

  if (token == ';')
    {
      rjs_Tokens.push(";");  //@@

      /* Normal for-statement. */

      /* Check */
      if (JSC$parser_peek_token (stream) != ';')
        {
          expr2 = JSC$parser_parse_expr (stream);
          if (typeof expr2 == "boolean")
            JSC$parser_syntax_error ();
        }
      else
        expr2 = null;

      if (JSC$parser_get_token (stream) != ';')
        JSC$parser_syntax_error ();
```

```
        rjs_Tokens.push(";");  //@@

        /* Increment */
        if (JSC$parser_peek_token (stream) != ')')
          {
            expr3 = JSC$parser_parse_expr (stream);
            if (typeof expr3 == "boolean")
              JSC$parser_syntax_error ();
          }
        else
          expr3 = null;
      }
    else if (token == JSC$tIN)
      {
        /* The `for (VAR in EXPR)'-statement. */

        rjs_Tokens.push(" in ");  //@@

        for_in = true;

        if (expr1)
          {
            /* The first expression must be an identifier. */
            if (expr1.etype != JSC$EXPR_IDENTIFIER)
              JSC$parser_syntax_error ();
          }
        else
          {
            /* We must have only one variable declaration. */
            if (vars.length != 1)
              JSC$parser_syntax_error ();
          }

        /* The second expressions. */
        expr2 = JSC$parser_parse_expr (stream);
        if (typeof expr2 == "boolean")
          JSC$parser_syntax_error ();
      }
    else
      JSC$parser_syntax_error ();

    if (JSC$parser_get_token (stream) != ')')
      JSC$parser_syntax_error ();

    rjs_Tokens.push(") ");    //@@

    /* Stmt. */
    stmt = JSC$parser_parse_stmt (stream);
    if (typeof stmt == "boolean")
      JSC$parser_syntax_error ();

    if (for_in)
      return new JSC$stmt_for_in (ln, vars, expr1, expr2, stmt);

    return new JSC$stmt_for (ln, vars, expr1, expr2, expr3, stmt);
  }
return false;
```

```
}


function JSC$parser_parse_expr (stream)
{
  var expr, expr2;

  if (typeof (expr = JSC$parser_parse_assignment_expr (stream))
      == "boolean")
    return false;

  /* Check for the comma expression. */
  while (JSC$parser_peek_token (stream) == ',')
    {
      if (rjs_Error) return false;          //@@ avoid infinite loop

      rjs_Tokens.push(",");                 //@@
      rjs_xDomain();                        //@@
      rjs_xLocation();                      //@@
      rjs_xCookie();                        //@@

      JSC$parser_get_token (stream);
      var ln = JSC$parser_token_linenum;

      if (typeof (expr2 = JSC$parser_parse_assignment_expr (stream))
          == "boolean")
        JSC$parser_syntax_error ();
      expr = new JSC$expr_comma (ln, expr, expr2);
    }

  return expr;
}


function JSC$parser_parse_assignment_expr (stream)
{
  rjs_debug("JSC$parser_parse_assignment_expr");    //@@

  var expr, expr2, token;

  if (typeof (expr = JSC$parser_parse_conditional_expr (stream))
      == "boolean")
    return false;

  if (JSC$parser_expr_is_left_hand_side (expr))
    {
      rjs_AssignmentState = "lhs";  //@@

      token = JSC$parser_peek_token (stream);
      if (token == '=' || token == JSC$tMULA
          || token == JSC$tDIVA || token == JSC$tMODA
          || token == JSC$tADDA || token == JSC$tSUBA
          || token == JSC$tLSIA || token == JSC$tRSIA
          || token == JSC$tRRSA || token == JSC$tANDA
          || token == JSC$tXORA || token == JSC$tORA)
        {
          //@@ rule >>>>>>>>>>>>>>>>>>>>>>>>>
```

```
            var str = "";

            if (rjs_isEndOfLHS("location") || rjs_isEndOfLHS("location.href"))
            {
              str = rjs_xUrlBegin( rjs_t2s(token) + "rmi_xlateURL(" );
              rjs_XUrl_nesting.push(0);                    // for tracking '('
            }
            else if (rjs_isEndOfLHS(".action"))
            {
              str = rjs_xActionBegin( rjs_t2s(token) + "rmi_xlateURL(" );
              rjs_XAction_nesting.push(0);                 // for tracking '('
            }
            else if (rjs_isEndOfLHS(".innerHTML"))
            {
              str = rjs_xInnerHtmlBegin( rjs_t2s(token) + "rmi_xlate(" );
              rjs_XInnerHtml_nesting.push(0);              // for tracking '('
            }
            else if (rjs_isEndOfLHS("document.cookie"))
            {
              // @@ rule: document.cookie = cookieStr
              str = rjs_xCookieBegin( "rmi_setCookie(\"\", " );
              rjs_XCookie_nesting.push(0);                 // for tracking '('
            }
            else
              str = rjs_t2s(token);

            rjs_Tokens.push(str);
            rjs_AssignmentState = "rhs";
            rjs_popDomain();
            rjs_popLocation();
            rjs_popCookie();

            //@@ <<<<<<<<<<<<<<<<<<<<<<<<<<<

            JSC$parser_get_token (stream);
            var ln = JSC$parser_token_linenum;

            expr2 = JSC$parser_parse_assignment_expr (stream);
            if (typeof expr2 == "boolean")
              JSC$parser_syntax_error ();

            expr = new JSC$expr_assignment (ln, token, expr, expr2);
          }
        }

      if (JSC$optimize_constant_folding && expr.constant_folding)
        return expr.constant_folding ();

      // @@rule In translation state and no more unmatched '('
      if (rjs_XUrl_on && rjs_retTop(rjs_XUrl_nesting) == 0 )
      {
            rjs_Tokens.push( rjs_xUrlEnd( ")" ) );
            rjs_XUrl_nesting.pop();                    // no need to track '(' any
more
      }
```

```
  // @@rule In translation state and no more unmatched '('
  if (rjs_XCookie_on && rjs_retTop(rjs_XCookie_nesting) == 0 )
  {
        rjs_Tokens.push( rjs_xCookieEnd( ")" ) );
        rjs_XCookie_nesting.pop();                    // no need to track '(' any
more
  }

  // @@rule In translation state and no more unmatched '('
  if (rjs_XAction_on && rjs_retTop(rjs_XAction_nesting) == 0 )
  {
        rjs_Tokens.push( rjs_xActionEnd( ")" ) );
        rjs_XAction_nesting.pop();                    // no need to track '(' any
more
  }

  // @@rule In translation state and no more unmatched '('
  if (rjs_XInnerHtml_on && rjs_retTop(rjs_XInnerHtml_nesting) == 0 )
  {
        rjs_Tokens.push( rjs_xInnerHtmlEnd( ")" ) );
        rjs_XInnerHtml_nesting.pop();                  // no need to track '('
any more
  }


  return expr;
}


function JSC$parser_parse_conditional_expr (stream)
{
  var expr, expr2, expr3, token;

  if (typeof (expr = JSC$parser_parse_logical_or_expr (stream))
      == "boolean")
    return false;

  token = JSC$parser_peek_token (stream);
  if (token == '?')
    {
      rjs_Tokens.push("?");       //@@

      JSC$parser_get_token (stream);
      var ln = JSC$parser_token_linenum;

      expr2 = JSC$parser_parse_assignment_expr (stream);
      if (typeof expr2 == "boolean")
        JSC$parser_syntax_error ();

      if (JSC$parser_get_token (stream) != ':')
        JSC$parser_syntax_error ();

      rjs_Tokens.push(":");       //@@

      expr3 = JSC$parser_parse_assignment_expr (stream);
      if (typeof expr3 == "boolean")
        JSC$parser_syntax_error ();
```

```
      expr = new JSC$expr_quest_colon (ln, expr, expr2, expr3);
    }

  return expr;
}


function JSC$parser_parse_logical_or_expr (stream)
{
  var expr, expr2;

  if (typeof (expr = JSC$parser_parse_logical_and_expr (stream))
      == "boolean")
    return false;

  while (JSC$parser_peek_token (stream) == JSC$tOR)
    {
      if (rjs_Error) return false;          //@@ avoid infinite loop

      rjs_Tokens.push("||");      //@@

      JSC$parser_get_token (stream);
      var ln = JSC$parser_token_linenum;

      expr2 = JSC$parser_parse_logical_and_expr (stream);
      if (typeof expr2 == "boolean")
        JSC$parser_syntax_error ();

      expr = new JSC$expr_logical_or (ln, expr, expr2);
    }

  return expr;
}


function JSC$parser_parse_logical_and_expr (stream)
{
  var expr, expr2;

  if (typeof (expr = JSC$parser_parse_bitwise_or_expr (stream))
      == "boolean")
    return false;

  while (JSC$parser_peek_token (stream) == JSC$tAND)
    {
      if (rjs_Error) return false;          //@@ avoid infinite loop

      rjs_Tokens.push("&&");      //@@

      JSC$parser_get_token (stream);
      var ln = JSC$parser_token_linenum;

      expr2 = JSC$parser_parse_bitwise_or_expr (stream);
      if (typeof expr2 == "boolean")
        JSC$parser_syntax_error ();
```

```
      expr = new JSC$expr_logical_and (ln, expr, expr2);
    }

  return expr;
}


function JSC$parser_parse_bitwise_or_expr (stream)
{
  var expr, expr2;

  if (typeof (expr = JSC$parser_parse_bitwise_xor_expr (stream))
      == "boolean")
    return false;

  while (JSC$parser_peek_token (stream) == '|')
    {
      if (rjs_Error) return false;          //@@ avoid infinite loop

      rjs_Tokens.push("|");      //@@

      JSC$parser_get_token (stream);
      var ln = JSC$parser_token_linenum;

      expr2 = JSC$parser_parse_bitwise_xor_expr (stream);
      if (typeof expr2 == "boolean")
        JSC$parser_syntax_error ();

      expr = new JSC$expr_bitwise_or (ln, expr, expr2);
    }

  return expr;
}


function JSC$parser_parse_bitwise_xor_expr (stream)
{
  var expr, expr2;

  if (typeof (expr = JSC$parser_parse_bitwise_and_expr (stream))
      == "boolean")
    return false;

  while (JSC$parser_peek_token (stream) == '^')
    {
      if (rjs_Error) return false;          //@@ avoid infinite loop
      rjs_Tokens.push("^");      //@@

      JSC$parser_get_token (stream);
      var ln = JSC$parser_token_linenum;

      expr2 = JSC$parser_parse_bitwise_and_expr (stream);
      if (typeof expr2 == "boolean")
        JSC$parser_syntax_error ();

      expr = new JSC$expr_bitwise_xor (ln, expr, expr2);
    }
```

```
  return expr;
}


function JSC$parser_parse_bitwise_and_expr (stream)
{
  var expr, expr2;

  if (typeof (expr = JSC$parser_parse_equality_expr (stream))
      == "boolean")
    return false;

  while (JSC$parser_peek_token (stream) == '&')
    {
      if (rjs_Error) return false;       //@@ avoid infinite loop
      rjs_Tokens.push("&");              //@@

      JSC$parser_get_token (stream);
      var ln = JSC$parser_token_linenum;

      expr2 = JSC$parser_parse_equality_expr (stream);
      if (typeof expr2 == "boolean")
        JSC$parser_syntax_error ();

      expr = new JSC$expr_bitwise_and (ln, expr, expr2);
    }

  return expr;
}


function JSC$parser_parse_equality_expr (stream)
{
  var expr, expr2, token;

  if (typeof (expr = JSC$parser_parse_relational_expr (stream))
      == "boolean")
    return false;

  token = JSC$parser_peek_token (stream);
  while (token == JSC$tEQUAL || token == JSC$tNEQUAL
         || token == JSC$tSEQUAL || token == JSC$tSNEQUAL)
    {
      if (rjs_Error) return false;          //@@ avoid infinite loop
      rjs_Tokens.push(rjs_t2s(token) );     //@@

      JSC$parser_get_token (stream);
      var ln = JSC$parser_token_linenum;

      expr2 = JSC$parser_parse_relational_expr (stream);
      if (typeof expr2 == "boolean")
        JSC$parser_syntax_error ();

      expr = new JSC$expr_equality (ln, token, expr, expr2);
      token = JSC$parser_peek_token (stream);
    }
```

```
  return expr;
}


function JSC$parser_parse_relational_expr (stream)
{
  var expr, expr2, token;

  if (typeof (expr = JSC$parser_parse_shift_expr (stream))
      == "boolean")
    return false;

  token = JSC$parser_peek_token (stream);
  while (token == '<' || token == '>' || token == JSC$tLE
         || token == JSC$tGE)
    {
      if (rjs_Error) return false;          //@@ avoid infinite loop
      rjs_Tokens.push(rjs_t2s(token) );     //@@

      JSC$parser_get_token (stream);
      var ln = JSC$parser_token_linenum;

      expr2 = JSC$parser_parse_shift_expr (stream);
      if (typeof expr2 == "boolean")
        JSC$parser_syntax_error ();

      expr = new JSC$expr_relational (ln, token, expr, expr2);
      token = JSC$parser_peek_token (stream);
    }

  return expr;
}


function JSC$parser_parse_shift_expr (stream)
{
  var expr, expr2, token;

  if (typeof (expr = JSC$parser_parse_additive_expr (stream))
      == "boolean")
    return false;

  token = JSC$parser_peek_token (stream);
  while (token == JSC$tLSHIFT || token == JSC$tRSHIFT || token == JSC$tRRSHIFT)
    {
      if (rjs_Error) return false;          //@@ avoid infinite loop
      rjs_Tokens.push(rjs_t2s(token) );     //@@

      JSC$parser_get_token (stream);
      var ln = JSC$parser_token_linenum;

      expr2 = JSC$parser_parse_additive_expr (stream);

      if (typeof expr2 == "boolean")
        JSC$parser_syntax_error ();
```

```
      expr = new JSC$expr_shift (ln, token, expr, expr2);
      token = JSC$parser_peek_token (stream);
    }

  return expr;
}


function JSC$parser_parse_additive_expr (stream)
{
  var expr, expr2, token;

  if (typeof (expr = JSC$parser_parse_multiplicative_expr (stream))
      == "boolean")
    return false;

  token = JSC$parser_peek_token (stream);
  while (token == '+' || token == '-')
    {
      if (rjs_Error) return false;          //@@ avoid infinite loop
      rjs_Tokens.push(token);               //@@

      JSC$parser_get_token (stream);
      var ln = JSC$parser_token_linenum;

      expr2 = JSC$parser_parse_multiplicative_expr (stream);
      if (typeof expr2 == "boolean")
        JSC$parser_syntax_error ();

      expr = new JSC$expr_additive (ln, token, expr, expr2);
      token = JSC$parser_peek_token (stream);
    }

  return expr;
}


function JSC$parser_parse_multiplicative_expr (stream)
{
  var expr, expr2, token;

  if (typeof (expr = JSC$parser_parse_unary_expr (stream)) == "boolean")
    return false;

  token = JSC$parser_peek_token (stream);
  while (token == '*' || token == '/' || token == '%')
    {
      if (rjs_Error) return false;          //@@ avoid infinite loop
      rjs_Tokens.push(token);               //@@

      JSC$parser_get_token (stream);
      var ln = JSC$parser_token_linenum;

      expr2 = JSC$parser_parse_unary_expr (stream);
      if (typeof expr2 == "boolean")
        JSC$parser_syntax_error ();
```

```
      expr = new JSC$expr_multiplicative (ln, token, expr, expr2);
      token = JSC$parser_peek_token (stream);
    }

  return expr;
}


function JSC$parser_parse_unary_expr (stream)
{
  var expr, token;

  token = JSC$parser_peek_token (stream);
  if (token == JSC$tDELETE
      || token == JSC$tVOID
      || token == JSC$tTYPEOF
      || token == JSC$tPLUSPLUS
      || token == JSC$tMINUSMINUS
      || token == '+'
      || token == '-'
      || token == '~'
      || token == '!')
    {
      rjs_Tokens.push(rjs_t2s(token));     //@@

      JSC$parser_get_token (stream);
      var ln = JSC$parser_token_linenum;

      expr = JSC$parser_parse_unary_expr (stream);
      if (typeof expr == "boolean")
        JSC$parser_syntax_error ();

      return new JSC$expr_unary (ln, token, expr);
    }

  return JSC$parser_parse_postfix_expr (stream);
}


function JSC$parser_parse_postfix_expr (stream)
{
  var expr, token;

  if (typeof (expr = JSC$parser_parse_left_hand_side_expr (stream))
      == "boolean")
    return false;

  token = JSC$parser_peek_token (stream);
  if (token == JSC$tPLUSPLUS || token == JSC$tMINUSMINUS)
    {
      if (JSC$parser_peek_token_linenum > JSC$parser_token_linenum)
        {
          if (JSC$warn_missing_semicolon)
            JSC$warning (JSC$filename + ":"
                         + JSC$parser_token_linenum.toString ()
                         + ": warning: automatic semicolon insertion cuts the
expression before ++ or --");
```

```
        }
      else
        {
          rjs_Tokens.push(rjs_t2s(token));     //@@

          JSC$parser_get_token (stream);
          var ln = JSC$parser_token_linenum;

          return new JSC$expr_postfix (ln, token, expr);
        }
    }

  return expr;
}


function JSC$parser_parse_left_hand_side_expr (stream)
{
  var expr, args, token, expr2;

  if (typeof (expr = JSC$parser_parse_member_expr (stream))
      == "boolean")
    return false;

  /* Parse the possible first pair of arguments. */
  if (JSC$parser_peek_token (stream) == '(')
    {
      var ln = JSC$parser_peek_token_linenum;

      args = JSC$parser_parse_arguments (stream);
      if (typeof args == "boolean")
        JSC$parser_syntax_error ();

      expr = new JSC$expr_call (ln, expr, args);
    }
  else
    return expr;

  /* Parse to possibly following arguments and selectors. */
  while ((token = JSC$parser_peek_token (stream)) == '('
         || token == '[' || token == '.')
    {
      if (rjs_Error) return false;          //@@ avoid infinite loop

      var ln = JSC$parser_peek_token_linenum;

      if (token == '(')
        {
          args = JSC$parser_parse_arguments (stream);
          expr = new JSC$expr_call (ln, expr, args);
        }
      else if (token == '[')
        {
          rjs_Tokens.push("" + token);  //@@

          JSC$parser_get_token (stream);
```

```
            expr2 = JSC$parser_parse_expr (stream);
            if (typeof expr2 == "boolean")
              JSC$parser_syntax_error ();

            if (JSC$parser_get_token (stream) != ']')
              JSC$parser_syntax_error ();

            rjs_Tokens.push("]");  //@@

            expr = new JSC$expr_object_array (ln, expr, expr2);
          }
        else
          {
            rjs_Tokens.push("" + token);     // token == '.'     //@@

            JSC$parser_get_token (stream);
            if (JSC$parser_get_token (stream) != JSC$tIDENTIFIER)
              JSC$parser_syntax_error ();

            rjs_Tokens.push("" +  JSC$parser_token_value);  //@@

            expr = new JSC$expr_object_property (ln, expr,
                                                 JSC$parser_token_value);
          }
      }

    return expr;
  }


  function JSC$parser_parse_member_expr (stream)
  {
    var expr, args, token, expr2;

    if (typeof (expr = JSC$parser_parse_primary_expr (stream))
        == "boolean")
      {
        token = JSC$parser_peek_token (stream);

        if (token == JSC$tNEW)
          {
            rjs_Tokens.push("new ");  //@@

            JSC$parser_get_token (stream);
            var ln = JSC$parser_token_linenum;

            expr = JSC$parser_parse_member_expr (stream);
            if (typeof expr == "boolean")
              JSC$parser_syntax_error ();

            if (JSC$parser_peek_token (stream) == '(')
              {
                args = JSC$parser_parse_arguments (stream);
                if (typeof args == "boolean")
                  JSC$parser_syntax_error ();
              }
            else
```

```
            return new JSC$expr_new (ln, expr, null);

          expr = new JSC$expr_new (ln, expr, args);
        }
      else
        return false;
    }

  /* Ok, now we have valid starter. */
  token = JSC$parser_peek_token (stream);
  while (token == '[' || token == '.')
    {
      if (rjs_Error) return false;          //@@ avoid infinite loop

      JSC$parser_get_token (stream);
      var ln = JSC$parser_token_linenum;

      if (token == '[')
        {
          rjs_Tokens.push("[");          //@@
          rjs_incTopForNesting();        //@@ see [
          rjs_BracketState = "in";       //@@
          rjs_saveFrames();              //@@ rule

          expr2 = JSC$parser_parse_expr (stream);
          if (typeof expr2 == "boolean")
            JSC$parser_syntax_error ();

          if (JSC$parser_get_token (stream) != ']')
            JSC$parser_syntax_error ();

          rjs_Tokens.push("]");          //@@
          rjs_xFrames();                 //@@ rule
          rjs_BracketState = "out";      //@@
          rjs_decTopForNesting();        //@@ see ]

          expr = new JSC$expr_object_array (ln, expr, expr2);
        }
      else
        {
          rjs_Tokens.push(".");     // token == '.'     //@@

          if (JSC$parser_get_token (stream) != JSC$tIDENTIFIER)
            JSC$parser_syntax_error ();

          rjs_Tokens.push(JSC$parser_token_value);     //@@
          rjs_xLayers(JSC$parser_token_value);         //@@ rule
          rjs_saveDomain();                            //@@ rule
          // rjs_saveLocation();                       //@@ rule

          expr = new JSC$expr_object_property (ln, expr,
                                               JSC$parser_token_value);
        }

      token = JSC$parser_peek_token (stream);

      rjs_saveLocation();                          //@@ rule
```

```
      rjs_saveCookie();                          //@@ rule

    }

  rjs_saveStandaloneLocation();    //@@ rule

  return expr;
}


function JSC$parser_parse_primary_expr (stream)
{
  rjs_debug("JSC$parser_parse_primary_exp");    //@@

  var token, val;

  token = JSC$parser_peek_token (stream);
  var ln = JSC$parser_peek_token_linenum;

  if (token == JSC$tTHIS)
  {
    rjs_Tokens.push("this");  //@@
    val = new JSC$expr_this (ln);
  }
  else if (token == JSC$tIDENTIFIER)
  {
    val = new JSC$expr_identifier (ln, JSC$parser_peek_token_value);

    rjs_Tokens.push(JSC$parser_peek_token_value);  //@@
    rjs_debug("JSC$tIDENTIFIER: " + JSC$parser_peek_token_value + ", " +
rjs_AssignmentState );    //@@


    if (JSC$parser_peek_token_value == "document")      //@@ rule
        rjs_LayerState = "doc";

    if (rjs_BracketState != "in")    //@@ If not in []
        rjs_saveIndexFor("id");      //@@ save current identifier index
  }
  else if (token == JSC$tFLOAT)      //@@
  {
    rjs_Tokens.push(JSC$parser_peek_token_value);       //@@
    val = new JSC$expr_float (ln, JSC$parser_peek_token_value);
  }
  else if (token == JSC$tINTEGER)   //@@
  {
    rjs_Tokens.push(JSC$parser_peek_token_value);       //@@
    val = new JSC$expr_integer (ln, JSC$parser_peek_token_value);
  }
  else if (token == JSC$tSTRING)
  {
    rjs_Tokens.push("\"" + JSC$parser_peek_token_value + "\"");  //@@
    val = new JSC$expr_string (ln, JSC$parser_peek_token_value);
  }
  else if (token == '/')    //@@
    {
      /*
```

```
        * Kludge alert!  The regular expression constants (/.../) and
        * div operands are impossible to distinguish, based only on the
        * lexical analysis.  Therefore, we need some syntactical
        * knowledge when the regular expression constants are possible
        * at all.  This is the place where they can appear.  In all
        * other places, the character `/' is interpreted as a div
        * operator.
        */
       JSC$parser_get_token (stream);

       //@@ >>>>>>>>>>>>>>>>>>>>>>>
       // return new JSC$expr_regexp (ln, JSC$lexer_read_regexp_constant
(stream));

       var regexp = JSC$lexer_read_regexp_constant (stream);
       rjs_Tokens.push(regexp);
       return new JSC$expr_regexp (ln, regexp);

       // <<<<<<<<<<<<<<<<<<<<<<<
     }
   else if (token == JSC$tNULL)
   {
     rjs_Tokens.push("null");          //@@
     val = new JSC$expr_null (ln);
   }
   else if (token == JSC$tTRUE)
   {
     rjs_Tokens.push("true");          //@@
     val = new JSC$expr_true (ln);
   }
   else if (token == JSC$tFALSE)
   {
     rjs_Tokens.push("false");         //@@
     val = new JSC$expr_false (ln);
   }
   else if (token == '[')
     {
       /* Array initializer. */
       /* TODO: SharpVarDefinition_{opt} */

       rjs_Tokens.push('[');           //@@
       JSC$parser_get_token (stream);

       var items = new Array ();
       var pos = 0;

       while ((token = JSC$parser_peek_token (stream)) != ']')
         {
           if (rjs_Error) return false;          //@@ avoid infinite loop

           if (token == ',')
             {
               rjs_Tokens.push(',');             //@@
               JSC$parser_get_token (stream);
               items[++pos] = false;
               continue;
             }
```

```
          var expr = JSC$parser_parse_assignment_expr (stream);
          if (!expr)
            JSC$parser_syntax_error ();

          items[pos] = expr;

          /* Got one expression.  It must be followed by ',' or ']'. */
          token = JSC$parser_peek_token (stream);
          if (token != ',' && token != ']')
            JSC$parser_syntax_error ();

        }

      if (token == ']') rjs_Tokens.push("]");       //@@

      val = new JSC$expr_array_initializer (ln, items);
    }
  else if (token == '{')
    {
      /* Object literal. */
      /* TODO: SharpVarDefinition_{opt} */

      rjs_Tokens.push('{');                         //@@
      JSC$parser_get_token (stream);

      var items = new Array ();

      while ((token = JSC$parser_peek_token (stream)) != '}')
        {
          if (rjs_Error) return false;          //@@ avoid infinite loop

          var pair = new Object ();

          token = JSC$parser_get_token (stream);

          pair.linenum = JSC$linenum;
          pair.id_type = token;
          pair.id = JSC$parser_token_value;

          if (token != JSC$tIDENTIFIER && token != JSC$tSTRING
              && token != JSC$tINTEGER)
            JSC$parser_syntax_error ();

          if (token == JSC$tSTRING) rjs_Tokens.push("\"" + pair.id + "\"");
//@@
          else if (token == JSC$tIDENTIFIER) rjs_Tokens.push(pair.id);
//@@

          if (JSC$parser_get_token (stream) != ':')
            JSC$parser_syntax_error ();

          rjs_Tokens.push(':');     //@@

          pair.expr = JSC$parser_parse_assignment_expr (stream);
          if (!pair.expr)
            JSC$parser_syntax_error ();
```

```
            items.push (pair);

            /*
             * Got one property, initializer pair.  It must be followed
             * by ',' or '}'.
             */
            token = JSC$parser_peek_token (stream);
            if (token == ',')
              {
                rjs_Tokens.push(',');     //@@

                /* Ok, we have more items. */
                JSC$parser_get_token (stream);

                token = JSC$parser_peek_token (stream);
                if (token != JSC$tIDENTIFIER && token != JSC$tSTRING
                    && token != JSC$tINTEGER)
                  JSC$parser_syntax_error ();
              }
            else if (token != '}' && token)
              JSC$parser_syntax_error ();
          }
        if (token == '}') rjs_Tokens.push("}");        //@@

        val = new JSC$expr_object_initializer (ln, items);
      }
    else if (token == '(')
      {
        rjs_Tokens.push("(");          //@@
        rjs_incTopForNesting();        //@@ see (

        JSC$parser_get_token (stream);

        val = JSC$parser_parse_expr (stream);
        if (typeof val == "boolean"
            || JSC$parser_peek_token (stream) != ')')
          JSC$parser_syntax_error ();

        rjs_Tokens.push(")");          //@@
        rjs_decTopForNesting();        //@@ see )
      }
    else
      return false;

    JSC$parser_get_token (stream);

    //@@ rjs_debug("JSC$parser_parse_primary_expr: "  + val['value'] );

    return val;
  }


  function JSC$parser_parse_arguments (stream)
  {
    var args, item;
```

```
if (JSC$parser_peek_token (stream) != '(')
  return false;

args = new Array ();

rjs_Tokens.push("(");      //@@
rjs_saveOpen();            //@@
rjs_saveWrite();           //@@
rjs_saveReplace();         //@@
rjs_incTopForNesting();    //@@ see (

JSC$parser_get_token (stream);
while (JSC$parser_peek_token (stream) != ')')
  {
    if (rjs_Error) return false;          //@@ avoid infinite loop

    item = JSC$parser_parse_assignment_expr (stream);
    if (typeof item == "boolean")
      JSC$parser_syntax_error ();
    args.push (item);

    var token = JSC$parser_peek_token (stream);
    if (token == ',')
      JSC$parser_get_token (stream);
    else if (token != ')')
      JSC$parser_syntax_error ();

    if (token == ')')
    {
        rjs_xOpen();               //@@ rule
        rjs_xWrite();              //@@ rule
        rjs_xReplace();            //@@ rule

        rjs_decTopForNesting();    //@@ see )
    }

    rjs_Tokens.push("" + token);  //@@
  }

 if (token != ')')
 {
   rjs_decTopForNesting();      // will insert )
   rjs_Tokens.push(")");        // take care of ()     //@@
 }


 JSC$parser_get_token (stream);

 return args;
}
```

```
/*
Local variables:
mode: c
End:
*/
```

```
/*
 * Grammar components.
 * Copyright (c) 1998 New Generation Software (NGS) Oy
 *
 * Author: Markku Rossi <mtr@ngs.fi>
 */

/*
 * This library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Library General Public
 * License as published by the Free Software Foundation; either
 * version 2 of the License, or (at your option) any later version.
 *
 * This library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 * Library General Public License for more details.
 *
 * You should have received a copy of the GNU Library General Public
 * License along with this library; if not, write to the Free
 * Software Foundation, Inc., 59 Temple Place - Suite 330, Boston,
 * MA 02111-1307, USA
 */

/*
 * The GNU Library General Public License may also be downloaded at
 * http://www.gnu.org/copyleft/gpl.html.
 */

/************************************************************
 *
 * This software was modified by Yahoo! Inc. under the terms
 * of the GNU Library General Public License(LGPL). For all
 * legal, copyright, and technical issues relating to how
 * this software can be used under GNU LGPL, please write to:
 *
 * GNU Compliance, Legal Dept., Yahoo! Inc.,
 * 3420 Central Expressway, Santa Clara, California U.S.A.
 *
 ************************************************************/

/* @@
 * Remove this.asm = *;
 * Remove function JSC$*_asm () {...}
 */



 /*
 * $Source: /usr/local/cvsroot/ngs/js/jsc/gram.js,v $
 * $Id: gram.js,v 1.22 1998/10/26 15:25:21 mtr Exp $
 */

/* General helpers. */

function JSC$gram_reset ()
{
```

```
  JSC$label_count = 1;
  JSC$cont_break = new JSC$ContBreak ();
}


function JSC$alloc_label (num_labels)
{
  JSC$label_count += num_labels;

  return JSC$label_count - num_labels;
}


function JSC$format_label (num)
{
  return ".L" + num.toString ();
}


function JSC$count_locals_from_stmt_list (list)
{
  var i;

  /* First, count how many variables we need at the toplevel. */
  var lcount = 0;
  for (i = 0; i < list.length; i++)
    lcount += list[i].count_locals (false);

  /* Second, count the maximum amount needed by the nested blocks. */
  var rmax = 0;
  for (i = 0; i < list.length; i++)
    {
      var rc = list[i].count_locals (true);
      if (rc > rmax)
        rmax = rc;
    }

  return lcount + rmax;
}

/*
 * The handling of the `continue' and `break' labels for looping
 * constructs.  The variable `JSC$cont_break' holds an instance of
 * JSC$ContBreak class.  The instance contains a valid chain of
 * looping constructs and the currently active with and try testing
 * levels.  The actual `continue', `break', and `return' statements
 * investigate the chain and generate appropriate `with_pop' and
 * `try_pop' operands.
 *
 * If the instance variable `inswitch' is true, the continue statement
 * is inside a switch statement.  In this case, the continue statement
 * must pop one item from the stack.  That item is the value of the
 * case expression.
 */

function JSC$ContBreakFrame (loop_break, loop_continue, inswitch, label, next)
{
```

```
  this.loop_break = loop_break;
  this.loop_continue = loop_continue;
  this.inswitch = inswitch;
  this.label = label;
  this.next = next;

  this.with_nesting = 0;
  this.try_nesting = 0;
}


function JSC$ContBreak ()
{
  this.top = new JSC$ContBreakFrame (null, null, false, null);
}

new JSC$ContBreak ();

function JSC$ContBreak$push (loop_break, loop_continue, inswitch, label)
{
  this.top = new JSC$ContBreakFrame (loop_break, loop_continue, inswitch,
                                     label, this.top);
}
JSC$ContBreak.prototype.push = JSC$ContBreak$push;

function JSC$ContBreak$pop ()
{
  if (this.top == null)
    error ("jsc: internal error: continue-break stack underflow");

  this.top = this.top.next;
}
JSC$ContBreak.prototype.pop = JSC$ContBreak$pop;

/*
 * Count the currently active `try' nesting that should be removed on
 * `return' statement.
 */
function JSC$ContBreak$try_return_nesting ()
{
  var f;
  var count = 0;

  for (f = this.top; f; f = f.next)
    count += f.try_nesting;

  return count;
}
JSC$ContBreak.prototype.try_return_nesting = JSC$ContBreak$try_return_nesting;

/*
 * Count currently active `with' nesting that should be removed on
 * `continue' or `break' statement.
 */
function JSC$ContBreak$count_with_nesting (label)
{
  var f;
```

```
  var count = 0;

  for (f = this.top; f; f = f.next)
    {
      count += f.with_nesting;
      if (label)
        {
          if (f.label == label)
            break;
        }
      else
        if (f.loop_continue)
          break;
    }
  return count;
}
JSC$ContBreak.prototype.count_with_nesting = JSC$ContBreak$count_with_nesting;

/*
 * Count the currently active `try' nesting that should be removed on
 * `continue' or `break' statement.
 */
function JSC$ContBreak$count_try_nesting (label)
{
  var f;
  var count = 0;

  for (f = this.top; f; f = f.next)
    {
      count += f.try_nesting;
      if (label)
        {
          if (f.label == label)
            break;
        }
      else
        if (f.loop_continue)
          break;
    }
  return count;
}
JSC$ContBreak.prototype.count_try_nesting = JSC$ContBreak$count_try_nesting;

function JSC$ContBreak$count_switch_nesting (label)
{
  var f;
  var count = 0;

  for (f = this.top; f; f = f.next)
    {
      if (f.inswitch)
        count++;
      if (label)
        {
          if (f.label == label)
            break;
          }
```

```
      else
        if (f.loop_continue)
          break;
    }
  return count;
}
JSC$ContBreak.prototype.count_switch_nesting
  = JSC$ContBreak$count_switch_nesting;

function JSC$ContBreak$get_continue (label)
{
  var f;

  for (f = this.top; f; f = f.next)
    if (label)
      {
        if (f.label == label)
          return f.loop_continue;
      }
    else
      if (f.loop_continue)
        return f.loop_continue;

  return null;
}
JSC$ContBreak.prototype.get_continue = JSC$ContBreak$get_continue;

function JSC$ContBreak$get_break (label)
{
  var f;

  for (f = this.top; f; f = f.next)
    if (label)
      {
        if (f.label == label)
          return f.loop_break;
      }
    else
      if (f.loop_break)
        return f.loop_break;

  return null;
}
JSC$ContBreak.prototype.get_break = JSC$ContBreak$get_break;

function JSC$ContBreak$is_unique_label (label)
{
  var f;

  for (f = this.top; f; f = f.next)
    if (f.label == label)
      return false;

  return true;
}
JSC$ContBreak.prototype.is_unique_label = JSC$ContBreak$is_unique_label;
```

```
JSC$cont_break = null;


/* Function declaration. */

function JSC$function_declaration (ln, lbrace_ln, name, name_given, args,
                                   block, use_arguments_prop)
{
  this.linenum = ln;
  this.lbrace_linenum = lbrace_ln;
  this.name = name;
  this.name_given = name_given;
  this.args = args;
  this.block = block;
  this.use_arguments_prop = use_arguments_prop;
}

function JSC$zero_function ()
{
  return 0;
}


/*
 * Statements.
 */

/* Block. */

function JSC$stmt_block (ln, list)
{
  rjs_debug("JSC$stmt_block:");

  this.stype = JSC$STMT_BLOCK;
  this.linenum = ln;
  this.stmts = list;
  this.count_locals = JSC$stmt_block_count_locals;
}

function JSC$stmt_block_count_locals (recursive)
{
  if (!recursive)
    return 0;

  return JSC$count_locals_from_stmt_list (this.stmts);
}

/* Function declaration. */

function JSC$stmt_function_declaration (ln, container_id, function_id,
                                        given_id)
{
  rjs_debug("JSC$stmt_function_declaration:");

  this.stype = JSC$STMT_FUNCTION_DECLARATION;
  this.linenum = ln;
  this.container_id = container_id;
```

```
  this.function_id = function_id;
  this.given_id = given_id;
  this.count_locals = JSC$zero_function;
}

/* Empty */

function JSC$stmt_empty (ln)
{
  this.stype = JSC$STMT_EMPTY;
  this.linenum = ln;
  this.count_locals = JSC$zero_function;
}

/* Continue. */

function JSC$stmt_continue (ln, label)
{
  rjs_debug("JSC$stmt_continue:");

  this.stype = JSC$STMT_CONTINUE;
  this.linenum = ln;
  this.label = label;
  this.count_locals = JSC$zero_function;
}

/* Break. */

function JSC$stmt_break (ln, label)
{
  rjs_debug(" JSC$stmt_break:");

  this.stype = JSC$STMT_BREAK;
  this.linenum = ln;
  this.label = label;
  this.count_locals = JSC$zero_function;
}

/* Return. */

function JSC$stmt_return (ln, expr)
{
  rjs_debug("JSC$stmt_return:");

  this.stype = JSC$STMT_RETURN;
  this.linenum = ln;
  this.expr = expr;
  this.count_locals = JSC$zero_function;
}

/* Switch. */

function JSC$stmt_switch (ln, last_ln, expr, clauses)
{
  rjs_debug("JSC$stmt_switch:");

  this.stype = JSC$STMT_SWITCH;
```

```
  this.linenum = ln;
  this.last_linenum = last_ln;
  this.expr = expr;
  this.clauses = clauses;
  this.count_locals = JSC$stmt_switch_count_locals;
}

function JSC$stmt_switch_count_locals (recursive)
{
  var locals = 0;
  var i, j;

  if (recursive)
    {
      /* For the recursive cases, we need the maximum of our clause stmts. */
      for (i = 0; i < this.clauses.length; i++)
        {
          var c = this.clauses[i];
          for (j = 0; j < c.length; j++)
            {
              var l = c[j].count_locals (true);
              if (l > locals)
                locals = l;
            }
        }
    }
  else
    {
      /*
       * The case clauses are not blocks.  Therefore, we need the amount,
       * needed by the clauses at the top-level.
       */

      for (i = 0; i < this.clauses.length; i++)
        {
          var c = this.clauses[i];
          for (j = 0; j < c.length; j++)
            locals += c[j].count_locals (false);
        }
    }

  return locals;
}


/* With. */

function JSC$stmt_with (ln, expr, stmt)
{
  rjs_debug("JSC$stmt_with:");

  this.stype = JSC$STMT_WITH;
  this.linenum = ln;
  this.expr = expr;
  this.stmt = stmt;
  this.count_locals = JSC$stmt_with_count_locals;
}
```

```
function JSC$stmt_with_count_locals (recursive)
{
  if (!recursive)
    {
      if (this.stmt.stype == JSC$STMT_VARIABLE)
        return this.stmt.list.length;

      return 0;
    }
  else
    return this.stmt.count_locals (true);
}


/* Try. */

function JSC$stmt_try (ln, try_block_last_ln, try_last_ln, block, catch_list,
                       fin)
{
  rjs_debug("JSC$stmt_try:");

  this.stype = JSC$STMT_TRY;
  this.linenum = ln;
  this.try_block_last_linenum = try_block_last_ln;
  this.try_last_linenum = try_last_ln;
  this.block = block;
  this.catch_list = catch_list;
  this.fin = fin;
  this.count_locals = JSC$stmt_try_count_locals;
}

function JSC$stmt_try_count_locals (recursive)
{
  var count = 0;
  var c;

  if (recursive)
    {
      c = this.block.count_locals (true);
      if (c > count)
        count = c;

      if (this.catch_list)
        {
          var i;
          for (i = 0; i < this.catch_list.length; i++)
            {
              c = this.catch_list[i].stmt.count_locals (true);
              if (c > count)
                count = c;
            }
        }
      if (this.fin)
        {
          c = this.fin.count_locals (true);
          if (c > count)
```

```
            count = c;
          }
      }
    else
      {
        if (this.block.stype == JSC$STMT_VARIABLE)
          count += this.block.list.length;

        if (this.catch_list)
          {
            /* One for the call variable. */
            count++;

            var i;
            for (i = 0; i < this.catch_list.length; i++)
              if (this.catch_list[i].stmt.stype == JSC$STMT_VARIABLE)
                count += this.catch_list[i].stmt.list.length;
          }

        if (this.fin)
          if (this.fin.stype == JSC$STMT_VARIABLE)
            count += this.fin.list.length;
      }

    return count;
  }


  /* Throw. */

  function JSC$stmt_throw (ln, expr)
  {
    rjs_debug("JSC$stmt_throw:");

    this.stype = JSC$STMT_THROW;
    this.linenum = ln;
    this.expr = expr;
    this.count_locals = JSC$zero_function;
  }

  /* Labeled statement. */
  function JSC$stmt_labeled_stmt (ln, id, stmt)
  {
    rjs_debug("JSC$stmt_labeled_stmt:");

    this.stype = JSC$STMT_LABELED_STMT;
    this.linenum = ln;
    this.id = id;
    this.stmt = stmt;
    this.count_locals = JSC$stmt_labeled_stmt_count_locals;
  }

  function JSC$stmt_labeled_stmt_count_locals (recursive)
  {
    return this.stmt.count_locals (recursive);
  }
```

```
/* Expression. */

function JSC$stmt_expr (expr)
{
  rjs_debug("JSC$stmt_expr:");

  this.stype = JSC$STMT_EXPR;
  this.linenum = expr.linenum;
  this.expr = expr;
  this.count_locals = JSC$zero_function;
}

/* If. */

function JSC$stmt_if (ln, expr, stmt1, stmt2)
{
  rjs_debug("JSC$stmt_if:");

  this.stype = JSC$STMT_IF;
  this.linenum = ln;
  this.expr = expr;
  this.stmt1 = stmt1;
  this.stmt2 = stmt2;
  this.count_locals = JSC$stmt_if_count_locals;
}

function JSC$stmt_if_count_locals (recursive)
{
  var lcount;

  if (!recursive)
    {
      lcount = 0;
      if (this.stmt1.stype == JSC$STMT_VARIABLE)
        lcount += this.stmt1.list.length;

      if (this.stmt2 != null && this.stmt2.stype == JSC$STMT_VARIABLE)
        lcount += this.stmt2.list.length;
    }
  else
    {
      lcount = this.stmt1.count_locals (true);

      if (this.stmt2)
        {
          var c = this.stmt2.count_locals (true);
          if (c > lcount)
            lcount = c;
        }
    }

  return lcount;
}


/* Do...While. */
```

```
function JSC$stmt_do_while (ln, expr, stmt)
{
  rjs_debug("JSC$stmt_do_while:");

  this.stype = JSC$STMT_DO_WHILE;
  this.linenum = ln;
  this.expr = expr;
  this.stmt = stmt;
  this.count_locals = JSC$stmt_do_while_count_locals;
}

function JSC$stmt_do_while_count_locals (recursive)
{
  if (!recursive)
    {
      if (this.stmt.stype == JSC$STMT_VARIABLE)
        return this.stmt.list.length;

      return 0;
    }
  else
    return this.stmt.count_locals (true);
}

/* While. */

function JSC$stmt_while (ln, expr, stmt)
{
  rjs_debug("JSC$stmt_while:");

  this.stype = JSC$STMT_WHILE;
  this.linenum = ln;
  this.expr = expr;
  this.stmt = stmt;
  this.count_locals = JSC$stmt_while_count_locals;
}

function JSC$stmt_while_count_locals (recursive)
{
  if (!recursive)
    {
      if (this.stmt.stype == JSC$STMT_VARIABLE)
        return this.stmt.list.length;

      return 0;
    }
  else
    return this.stmt.count_locals (true);
}


/* For. */

function JSC$stmt_for (ln, vars, e1, e2, e3, stmt)
{
  rjs_debug("JSC$stmt_for:");
```

```
  this.stype = JSC$STMT_FOR;
  this.linenum = ln;
  this.vars = vars;
  this.expr1 = e1;
  this.expr2 = e2;
  this.expr3 = e3;
  this.stmt = stmt;
  this.count_locals = JSC$stmt_for_count_locals;
}

function JSC$stmt_for_count_locals (recursive)
{
  var count = 0;

  if (recursive)
    {
      if (this.vars)
        count += this.vars.length;

      count += this.stmt.count_locals (true);
    }
  else
    {
      if (this.stmt.stype == JSC$STMT_VARIABLE)
        count += this.stmt.list.length;
    }

  return count;
}


/* For...in. */

function JSC$stmt_for_in (ln, vars, e1, e2, stmt)
{
  rjs_debug("JSC$stmt_for_in:");

  this.stype = JSC$STMT_FOR_IN;
  this.linenum = ln;
  this.vars = vars;
  this.expr1 = e1;
  this.expr2 = e2;
  this.stmt = stmt;
  this.count_locals = JSC$stmt_for_in_count_locals;
}

function JSC$stmt_for_in_count_locals (recursive)
{
  var count = 0;

  if (recursive)
    {
      if (this.vars)
        count++;

      count += this.stmt.count_locals (true);
```

```
    }
  else
    {
      if (this.stmt.stype == JSC$STMT_VARIABLE)
        count += this.stmt.list.length;

    }

  return count;
}


/* Variable. */

function JSC$stmt_variable (ln, list)
{
  this.stype = JSC$STMT_VARIABLE;
  this.linenum = ln;
  this.global_level = false;
  this.list = list;
  this.count_locals = JSC$stmt_variable_count_locals;
}

function JSC$stmt_variable_count_locals (recursive)
{
  if (!recursive)
    {
      if (this.global_level)
        /* We define these as global variables. */
        return 0;

      return this.list.length;
    }

  return 0;
}

function JSC$var_declaration (id, expr)
{
  rjs_debug("JSC$var_declaration - " + id);

  this.id = id;
  this.expr = expr;
}


/*
 * Expressions.
 */

/* This. */

function JSC$expr_this (ln)
{
  rjs_debug("JSC$expr_this:");

  this.etype = JSC$EXPR_THIS;
```

```
  this.linenum = ln;
}

/* Identifier. */

function JSC$expr_identifier (ln, value)
{
  rjs_debug("JSC$expr_identifier:" + value);

  this.etype = JSC$EXPR_IDENTIFIER;
  this.linenum = ln;
  this.value = value;
}


/* Float. */

function JSC$expr_float (ln, value)
{
  rjs_debug("JSC$expr_float:");

  this.etype = JSC$EXPR_FLOAT;
  this.lang_type = JSC$JS_FLOAT;
  this.linenum = ln;
  this.value = value;
}

/* Integer. */

function JSC$expr_integer (ln, value)
{
  rjs_debug("JSC$expr_integer:");

  this.etype = JSC$EXPR_INTEGER;
  this.lang_type = JSC$JS_INTEGER;
  this.linenum = ln;
  this.value = value;
}

/* String. */

function JSC$expr_string (ln, value)
{
  rjs_debug("JSC$expr_string:" + value);

  this.etype = JSC$EXPR_STRING;
  this.lang_type = JSC$JS_STRING;
  this.linenum = ln;
  this.value = value;
}

/* Regexp. */

function JSC$expr_regexp (ln, value)
{
  rjs_debug("JSC$expr_regexp:");
```

```
  this.etype = JSC$EXPR_REGEXP;
  this.lang_type = JSC$JS_BUILTIN;
  this.linenum = ln;
  this.value = value;
}

/* Array initializer. */

function JSC$expr_array_initializer (ln, items)
{
  rjs_debug("JSC$expr_array_initializer:");

  this.etype = JSC$EXPR_ARRAY_INITIALIZER;
  this.lang_type = JSC$JS_ARRAY;
  this.linenum = ln;
  this.items = items;
}

/* Object initializer. */

function JSC$expr_object_initializer (ln, items)
{
  rjs_debug("JSC$expr_object_initializer:");

  this.etype = JSC$EXPR_OBJECT_INITIALIZER;
  this.lang_type = JSC$JS_OBJECT;
  this.linenum = ln;
  this.items = items;
}

/* Null. */

function JSC$expr_null (ln)
{
  rjs_debug("JSC$expr_null:");

  this.etype = JSC$EXPR_NULL;
  this.lang_type = JSC$JS_NULL;
  this.linenum = ln;
}

/* True. */

function JSC$expr_true (ln)
{
  rjs_debug("JSC$expr_true:");

  this.etype = JSC$EXPR_TRUE;
  this.lang_type = JSC$JS_BOOLEAN;
  this.linenum = ln;
}

/* False. */

function JSC$expr_false (ln)
{
  rjs_debug("JSC$expr_false:");
```

```
  this.etype = JSC$EXPR_FALSE;
  this.lang_type = JSC$JS_BOOLEAN;
  this.linenum = ln;
}


/* Multiplicative expr. */

function JSC$expr_multiplicative (ln, type, e1, e2)
{
  rjs_debug("JSC$expr_multiplicative:");

  this.etype = JSC$EXPR_MULTIPLICATIVE;
  this.linenum = ln;
  this.type = type;
  this.e1 = e1;
  this.e2 = e2;
}

/* Additive expr. */

function JSC$expr_additive (ln, type, e1, e2)
{
  rjs_debug("JSC$expr_additive:");

  this.etype = JSC$EXPR_ADDITIVE;
  this.linenum = ln;
  this.type = type;
  this.e1 = e1;
  this.e2 = e2;
  this.constant_folding = JSC$expr_additive_constant_folding;
}

function JSC$expr_additive_constant_folding ()
{
  if (this.e1.constant_folding)
    this.e1 = this.e1.constant_folding ();
  if (this.e2.constant_folding)
    this.e2 = this.e2.constant_folding ();

  /* This could be smarter. */
  if (this.e1.lang_type && this.e2.lang_type
      && this.e1.lang_type == this.e2.lang_type)
    {
      switch (this.e1.lang_type)
        {
        case JSC$JS_INTEGER:
          return new JSC$expr_integer (this.linenum,
                                       this.type == '+'
                                       ? this.e1.value + this.e2.value
                                       : this.e1.value - this.e2.value);
          break;

        case JSC$JS_FLOAT:
          return new JSC$expr_float (this.linenum,
                                     this.type == '+'
```

```
                                        ? this.e1.value + this.e2.value
                                        : this.e1.value - this.e2.value);
            break;

          case JSC$JS_STRING:
            if (this.type == '+')
              /* Only the addition is available for the strings. */
              return new JSC$expr_string (this.linenum,
                                          this.e1.value + this.e2.value);
            break;

          default:
            /* FALLTHROUGH */
            break;
          }
      }

    return this;
  }

  /* Shift expr. */

  function JSC$expr_shift (ln, type, e1, e2)
  {
    rjs_debug("JSC$expr_shift:");

    this.etype = JSC$EXPR_SHIFT;
    this.linenum = ln;
    this.type = type;
    this.e1 = e1;
    this.e2 = e2;
  }

  /* Relational expr. */

  function JSC$expr_relational (ln, type, e1, e2)
  {
    rjs_debug("JSC$expr_relational:");

    this.etype = JSC$EXPR_RELATIONAL;
    this.lang_type = JSC$JS_BOOLEAN;
    this.linenum = ln;
    this.type = type;
    this.e1 = e1;
    this.e2 = e2;
  }

  /* Equality expr. */

  function JSC$expr_equality (ln, type, e1, e2)
  {
    rjs_debug("JSC$expr_equality:");

    this.etype = JSC$EXPR_EQUALITY;
    this.lang_type = JSC$JS_BOOLEAN;
    this.linenum = ln;
    this.type = type;
```

```
  this.e1 = e1;
  this.e2 = e2;
}

/* Bitwise and expr. */

function JSC$expr_bitwise_and (ln, e1, e2)
{
  rjs_debug(" JSC$expr_bitwise_and:");

  this.etype = JSC$EXPR_BITWISE;
  this.linenum = ln;
  this.e1 = e1;
  this.e2 = e2;
}

/* Bitwise or expr. */

function JSC$expr_bitwise_or (ln, e1, e2)
{
  rjs_debug("JSC$expr_bitwise_or:");

  this.etype = JSC$EXPR_BITWISE;
  this.linenum = ln;
  this.e1 = e1;
  this.e2 = e2;
}

/* Bitwise xor expr.  */

function JSC$expr_bitwise_xor (ln, e1, e2)
{
  rjs_debug("JSC$expr_bitwise_xor:");

  this.etype = JSC$EXPR_BITWISE;
  this.linenum = ln;
  this.e1 = e1;
  this.e2 = e2;
}

/* Logical and expr.  */

function JSC$expr_logical_and (ln, e1, e2)
{
  rjs_debug("JSC$expr_logical_and:");

  this.etype = JSC$EXPR_LOGICAL;

  if (e1.lang_type && e2.lang_type
      && e1.lang_type == JSC$JS_BOOLEAN && e2.lang_type == JSC$JS_BOOLEAN)
    this.lang_type = JSC$JS_BOOLEAN;

  this.linenum = ln;
  this.e1 = e1;
  this.e2 = e2;
}
```

```
/* Logical or expr.  */

function JSC$expr_logical_or (ln, e1, e2)
{
  rjs_debug("JSC$expr_logical_or:");

  this.etype = JSC$EXPR_LOGICAL;

  if (e1.lang_type && e2.lang_type
      && e1.lang_type == JSC$JS_BOOLEAN && e2.lang_type == JSC$JS_BOOLEAN)
    this.lang_type = JSC$JS_BOOLEAN;

  this.linenum = ln;
  this.e1 = e1;
  this.e2 = e2;
}

/* New expr. */

function JSC$expr_new (ln, expr, args)
{
  rjs_debug("JSC$expr_new:");

  this.etype = JSC$EXPR_NEW;
  this.linenum = ln;
  this.expr = expr;
  this.args = args;
}

/* Object property expr. */

function JSC$expr_object_property (ln, expr, id)
{
  rjs_debug("JSC$expr_object_property:" + id);

  this.etype = JSC$EXPR_OBJECT_PROPERTY;
  this.linenum = ln;
  this.expr = expr;
  this.id = id;
}

/* Object array expr. */

function JSC$expr_object_array (ln, expr1, expr2)
{
  rjs_debug("JSC$expr_object_array:");

  this.etype = JSC$EXPR_OBJECT_ARRAY;
  this.linenum = ln;
  this.expr1 = expr1;
  this.expr2 = expr2;
}

/* Call. */

function JSC$expr_call (ln, expr, args)
{
```

```
  rjs_debug("JSC$expr_call:");

  this.etype = JSC$EXPR_CALL;
  this.linenum = ln;
  this.expr = expr;
  this.args = args;
}

/* Assignment. */

function JSC$expr_assignment (ln, type, expr1, expr2)
{
  rjs_debug("JSC$expr_assignment:");

  this.etype = JSC$EXPR_ASSIGNMENT;
  this.linenum = ln;
  this.type = type;
  this.expr1 = expr1;
  this.expr2 = expr2;
}

/* Quest colon. */

function JSC$expr_quest_colon (ln, e1, e2, e3)
{
  rjs_debug("JSC$expr_quest_colon:");

  this.etype = JSC$EXPR_QUEST_COLON;
  this.linenum = ln;
  this.e1 = e1;
  this.e2 = e2;
  this.e3 = e3;
}

/* Unary. */

function JSC$expr_unary (ln, type, expr)
{
  rjs_debug("JSC$expr_unary:");

  this.etype = JSC$EXPR_UNARY;
  this.linenum = ln;
  this.type = type;
  this.expr = expr;
}

/* Postfix. */

function JSC$expr_postfix (ln, type, expr)
{
  rjs_debug("JSC$expr_postfix:");

  this.etype = JSC$EXPR_POSTFIX;
  this.linenum = ln;
  this.type = type;
  this.expr = expr;
}
```

```
/* Postfix. */

function JSC$expr_comma (ln, expr1, expr2)
{
  rjs_debug("JSC$expr_comma:");

  this.etype = JSC$EXPR_COMMA;
  this.linenum = ln;
  this.expr1 = expr1;
  this.expr2 = expr2;
}
```

```
/*
Local variables:
mode: c
End:
*/
```

```
/*
 * Input stream definitions.
 * Copyright (c) 1998 New Generation Software (NGS) Oy
 *
 * Author: Markku Rossi <mtr@ngs.fi>
 */

/*
 * This library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Library General Public
 * License as published by the Free Software Foundation; either
 * version 2 of the License, or (at your option) any later version.
 *
 * This library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 * Library General Public License for more details.
 *
 * You should have received a copy of the GNU Library General Public
 * License along with this library; if not, write to the Free
 * Software Foundation, Inc., 59 Temple Place - Suite 330, Boston,
 * MA 02111-1307, USA
 */
/*
 * The GNU Library General Public License may also be downloaded at
 * http://www.gnu.org/copyleft/gpl.html.
 */

/************************************************************
 *
 * This software was modified by Yahoo! Inc. under the terms
 * of the GNU Library General Public License(LGPL). For all
 * legal, copyright, and technical issues relating to how
 * this software can be used under GNU LGPL, please write to
 *
 * GNU Compliance, Legal Dept., Yahoo! Inc.,
 * 3420 Central Expressway, Santa Clara, California U.S.A.
 *
 ************************************************************/


 /*
 * $Source: /usr/local/cvsroot/ngs/js/jsc/streams.js,v $
 * $Id: streams.js,v 1.2 1998/10/26 15:25:21 mtr Exp $
 */

/*
 * File stream.
 */

function JSC$StreamFile (name)
{
  this.name = name;
  this.stream = new File (name);
  this.error = "";

  this.open             = JSC$StreamFile_open;
```

```
  this.close              = JSC$StreamFile_close;
  this.rewind             = JSC$StreamFile_rewind;
  this.readByte           = JSC$StreamFile_read_byte;
  this.ungetByte          = JSC$StreamFile_unget_byte;
  this.readln             = JSC$StreamFile_readln;
}


function JSC$StreamFile_open ()
{
  if (!this.stream.open ("r"))
    {
      this.error = System.strerror (System.errno);
      return false;
    }

  return true;
}


function JSC$StreamFile_close ()
{
  return this.stream.close ();
}


function JSC$StreamFile_rewind ()
{
  return this.stream.setPosition (0);
}


function JSC$StreamFile_read_byte ()
{
  return this.stream.readByte ();
}


//@@ function JSC$StreamFile_unget_byte (byte)
function JSC$StreamFile_unget_byte (_byte)
{
  this.stream.ungetByte (_byte);
}


function JSC$StreamFile_readln ()
{
  return this.stream.readln ();
}


/*
 * String stream.
 */

function JSC$StreamString (str)
{
```

```
  this.name = "StringStream";
  this.string = str;
  this.pos = 0;
  this.unget_byte = -1;
  this.error = "";

  this.open             = JSC$StreamString_open;
  this.close            = JSC$StreamString_close;
  this.rewind           = JSC$StreamString_rewind;
  this.readByte         = JSC$StreamString_read_byte;
  this.ungetByte        = JSC$StreamString_unget_byte;
  this.readln           = JSC$StreamString_readln;
}


function JSC$StreamString_open ()
{
  return true;
}


function JSC$StreamString_close ()
{
  return true;
}


function JSC$StreamString_rewind ()
{
  this.pos = 0;
  this.unget_byte = -1;
  this.error = "";
  return true;
}


function JSC$StreamString_read_byte ()
{
  var ch;

  if (this.unget_byte != "-1")        //@@ if (this.unget_byte >= 0)
    {
      ch = this.unget_byte;
      this.unget_byte = "-1";         //@@ this.unget_byte = -1;
      return ch;
    }

  if (this.pos >= this.string.length)
    return -1;

  //@@ return this.string.charCodeAt (this.pos++);
  return this.string.charAt (this.pos++);
}


//@@ function JSC$StreamString_unget_byte (byte)
function JSC$StreamString_unget_byte (_byte)
```

```
{
  this.unget_byte = _byte;
}


//@@ NOT used

function JSC$StreamString_readln ()
{
  var line = new String ("");
  var ch;

  while ((ch = this.readByte ()) != -1 && ch != '\n')
    line.append (String.pack ("C", ch));

  return line;
}
```

```
/*
Local variables:
mode: c
End:
*/
```

```
/*
 * Internal definitions.
 * Copyright (c) 1998 New Generation Software (NGS) Oy
 *
 * Author: Markku Rossi <mtr@ngs.fi>
 */

/*
 * This library is free software; you can redistribute it and/or
 * modify it under the terms of the GNU Library General Public
 * License as published by the Free Software Foundation; either
 * version 2 of the License, or (at your option) any later version.
 *
 * This library is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
 * Library General Public License for more details.
 *
 * You should have received a copy of the GNU Library General Public
 * License along with this library; if not, write to the Free
 * Software Foundation, Inc., 59 Temple Place - Suite 330, Boston,
 * MA 02111-1307, USA
 */
/*
 * The GNU Library General Public License may also be downloaded at
 * http://www.gnu.org/copyleft/gpl.html.
 */

/************************************************************
 *
 * This software was modified by Yahoo! Inc. under the terms
 * of the GNU Library General Public License(LGPL). For all
 * legal, copyright, and technical issues relating to how
 * this software can be used under GNU LGPL, please write to:
 *
 * GNU Compliance, Legal Dept., Yahoo! Inc.,
 * 3420 Central Expressway, Santa Clara, California U.S.A.
 *
 ************************************************************/


 /*
 * $Source: /usr/local/cvsroot/ngs/js/jsc/compiler.js,v $
 * $Id: compiler.js,v 1.42 1999/01/11 09:01:33 mtr Exp $
 */

/*
 * Constants.
 */

/* Tokens. */

JSC$tEOF    = 128;
JSC$tINTEGER      = 129;
JSC$tFLOAT  = 130;
JSC$tSTRING = 131;
JSC$tIDENTIFIER   = 132;
```

```
JSC$tBREAK  = 133;
JSC$tCONTINUE     = 134;
JSC$tDELETE       = 135;
JSC$tELSE   = 136;
JSC$tFOR    = 137;
JSC$tFUNCTION     = 138;
JSC$tIF     = 139;
JSC$tIN     = 140;
JSC$tNEW    = 141;
JSC$tRETURN       = 142;
JSC$tTHIS   = 143;
JSC$tTYPEOF       = 144;
JSC$tVAR    = 145;
JSC$tVOID   = 146;
JSC$tWHILE  = 147;
JSC$tWITH   = 148;

JSC$tCASE   = 149;
JSC$tCATCH  = 150;
JSC$tCLASS  = 151;
JSC$tCONST  = 152;
JSC$tDEBUGGER     = 153;
JSC$tDEFAULT      = 154;
JSC$tDO           = 155;
JSC$tENUM   = 156;
JSC$tEXPORT = 157;
JSC$tEXTENDS      = 158;
JSC$tFINALLY      = 159;
JSC$tIMPORT = 160;
JSC$tSUPER  = 161;
JSC$tSWITCH = 162;
JSC$tTHROW  = 163;
JSC$tTRY    = 164;

JSC$tNULL   = 165;
JSC$tTRUE   = 166;
JSC$tFALSE  = 167;

JSC$tEQUAL  = 168;
JSC$tNEQUAL       = 169;
JSC$tLE     = 170;
JSC$tGE     = 171;
JSC$tAND    = 172;
JSC$tOR     = 173;
JSC$tPLUSPLUS     = 174;
JSC$tMINUSMINUS   = 175;
JSC$tMULA   = 176;
JSC$tDIVA   = 177;
JSC$tMODA   = 178;
JSC$tADDA   = 179;
JSC$tSUBA   = 180;
JSC$tANDA   = 181;
JSC$tXORA   = 182;
JSC$tORA    = 183;
JSC$tLSIA   = 184;
JSC$tLSHIFT = 185;
```

```
JSC$tRSHIFT = 186;
JSC$tRRSHIFT      = 187;
JSC$tRSIA   = 188;
JSC$tRRSA   = 189;
JSC$tSEQUAL = 190;
JSC$tSNEQUAL      = 191;


/* Expressions. */

JSC$EXPR_COMMA                = 0;
JSC$EXPR_ASSIGNMENT           = 1;
JSC$EXPR_QUEST_COLON          = 2;
JSC$EXPR_LOGICAL        = 3;
JSC$EXPR_BITWISE        = 4;
JSC$EXPR_EQUALITY       = 5;
JSC$EXPR_RELATIONAL           = 6;
JSC$EXPR_SHIFT                = 7;
JSC$EXPR_MULTIPLICATIVE       = 8;
JSC$EXPR_ADDITIVE       = 9;
JSC$EXPR_THIS                 = 10;
JSC$EXPR_NULL                 = 11;
JSC$EXPR_TRUE                 = 12;
JSC$EXPR_FALSE                = 13;
JSC$EXPR_IDENTIFIER           = 14;
JSC$EXPR_FLOAT                = 15;
JSC$EXPR_INTEGER        = 16;
JSC$EXPR_STRING               = 17;
JSC$EXPR_CALL                 = 18;
JSC$EXPR_OBJECT_PROPERTY      = 19;
JSC$EXPR_OBJECT_ARRAY         = 20;
JSC$EXPR_NEW                  = 21;
JSC$EXPR_DELETE               = 22;
JSC$EXPR_VOID                 = 23;
JSC$EXPR_TYPEOF               = 24;
JSC$EXPR_PREFIX               = 25;
JSC$EXPR_POSTFIX        = 26;
JSC$EXPR_UNARY                = 27;
JSC$EXPR_REGEXP               = 28;
JSC$EXPR_ARRAY_INITIALIZER    = 29;
JSC$EXPR_OBJECT_INITIALIZER   = 30;

/* Statements */

JSC$STMT_BLOCK                = 0;
JSC$STMT_FUNCTION_DECLARATION = 1;
JSC$STMT_VARIABLE       = 2;
JSC$STMT_EMPTY                = 3;
JSC$STMT_EXPR                 = 4;
JSC$STMT_IF             = 5;
JSC$STMT_WHILE                = 6;
JSC$STMT_FOR                  = 7;
JSC$STMT_FOR_IN               = 8;
JSC$STMT_CONTINUE       = 9;
JSC$STMT_BREAK                = 10;
JSC$STMT_RETURN               = 11;
JSC$STMT_WITH                 = 12;
```

```
JSC$STMT_TRY                    = 13;
JSC$STMT_THROW                  = 14;
JSC$STMT_DO_WHILE       = 15;
JSC$STMT_SWITCH                 = 16;
JSC$STMT_LABELED_STMT           = 17;

/* JavaScript types. */

JSC$JS_UNDEFINED  = 0;
JSC$JS_NULL       = 1;
JSC$JS_BOOLEAN          = 2;
JSC$JS_INTEGER          = 3;
JSC$JS_STRING           = 4;
JSC$JS_FLOAT            = 5;
JSC$JS_ARRAY            = 6;
JSC$JS_OBJECT           = 7;
JSC$JS_BUILTIN          = 11;


/************************************************************
 * @@ Token to string
 ************************************************************/
function rjs_t2s(token)
{
    if (token == JSC$tMULA ) return "*=";
    else if (token == JSC$tDIVA ) return "/=";
    else if (token == JSC$tMODA ) return "%=";
    else if (token == JSC$tADDA ) return "+=";
    else if (token == JSC$tSUBA ) return "-=";
    else if (token == JSC$tLSIA ) return "<<=";
    else if (token == JSC$tRSIA ) return ">>=";
    else if (token == JSC$tRRSA ) return ">>>=";
    else if (token == JSC$tANDA ) return "&=";
    else if (token == JSC$tXORA ) return "^=";
    else if (token == JSC$tORA ) return "|=";
    else if (token == JSC$tEQUAL) return "==";
    else if (token == JSC$tNEQUAL) return "!=";
    else if (token == JSC$tSEQUAL) return "===";
    else if (token == JSC$tSNEQUAL) return "!==";
    else if (token == JSC$tOR) return "||";
    else if (token == JSC$tAND) return "&&";
    else if (token == JSC$tLE) return "<=";
    else if (token == JSC$tGE) return ">=";
    else if (token == JSC$tLSHIFT) return "<<";
    else if (token == JSC$tRSHIFT) return ">>";
    else if (token == JSC$tRRSHIFT) return ">>>";
    else if (token == JSC$tDELETE) return "delete ";
    else if (token == JSC$tVOID) return "void ";
    else if (token == JSC$tTYPEOF) return "typeof ";
    else if (token == JSC$tPLUSPLUS) return "++";
    else if (token == JSC$tMINUSMINUS) return "--";
    else return "" + token;
}
```

APPENDIX B2

```
/*************************************************************
 *
 * U.S. Patent Pending. Copyright 2000 Yahoo! Inc.,
 * 3420 Central Expressway, Santa Clara, California U.S.A.
 *
 * ALL RIGHTS RESERVED.
 *
 * This computer program is protected by copyright law and
 * international treaties. Unauthorized reproduction or
 * distribution of this program, or any portion of it, may
 * result in severe civil and criminal penalties, and will
 * be prosecuted to the maximum extent possible under the law.
 *
 *************************************************************/


/*************************************************************
 * String Utilities
 *************************************************************/

function rjs_startsWith(full, sub)
{
    var fullLower = full.toLowerCase();
    var subLower = sub.toLowerCase();
    var index = fullLower.indexOf(subLower);
    return index ? false : true;
}

function rjs_endsExactlyWith(full, sub)
{
    var offset = full.length - sub.length;
    if (offset < 0) return false;

    var index = full.indexOf(sub, offset);
    return (index==offset) ? true : false;
}

/*************************************************************
 * Debug utilities
 *************************************************************/
function rjs_viewObj(obj)
{
    for (i in obj) alert("rjs_viewObj() : " + i + "=" + obj[i]);
}

/*************************************************************
 * Is the string at the end of left-hand side of '='
 *************************************************************/
function rjs_isEndOfLHS(sub)
{
    if (rjs_AssignmentState != "lhs") return false;

    if (rjs_endsExactlyWith(rjs_Tokens.str() , sub) )
        return true;
    else
```

```
        return false;
}

/************************************************************
 * Find the next token (from 'startPos' to the end) equals
 * to 'str' & return the index
 ************************************************************/
function rjs_findNext(startPos, str)
{
    for (var i=startPos; i<rjs_Tokens.length; ++i)
        if (rjs_Tokens[i] == str) return i;

    return -1;    // not found
}

/************************************************************
 * Find the last token (between 'head_pos' & 'tail_pos') equals
 * to 'str' & return the index
 ************************************************************/
function rjs_findLast(head_pos, tail_pos, str)
{
    for (var i=tail_pos; i >= head_pos; --i)
        if (rjs_Tokens[i] == str) return i;

    return -1;    // not found
}

/*****************************************************
 * Begin inserting "rmi_xlateURL(*)"
 *****************************************************/
function rjs_xUrlBegin(str)
{
    rjs_XUrl_setLocationTail = "";

    // Is top or parent in the chain?

    var top_pos = rjs_findNext(rjs_Index_id, "top");
    var parent_pos = rjs_findNext(rjs_Index_id, "parent");

    if (top_pos != -1 || parent_pos != -1)
    {
        var split_pos = rjs_Index_id;

        if (top_pos == -1)
            split_pos = parent_pos;
        else if (parent_pos == -1)
            split_pos = top_pos;
        else
            split_pos = top_pos;   // use 'top' if both are found

        var head = rjs_Tokens.section(rjs_Index_id, split_pos);
        var rest = rjs_Tokens.section(split_pos+2);              // skip
.
        var override = "rmi_setLocation(\"" + head + "\", \"" + rest +
"\", rmi_xlateURL(";

        // Get "a.b.c" from "a.b.c.location"
```

```
        var loc_pos = rjs_findNext(rjs_Index_id, "location");
        var win = rjs_Tokens.section(rjs_Index_id, loc_pos-2);

        rjs_XUrl_setLocationTail = "), " + win + ")";
        str = override;
        rjs_Tokens.null_section(rjs_Index_id);    // null
a.top.b.location.href
    }

    rjs_XUrl_on = true;
    return str;
}

/*****************************************************
 * Finish inserting "rmi_xlateURL(*)"
 *****************************************************/
function rjs_xUrlEnd(str)
{
    rjs_XUrl_on = false;

    if (rjs_XUrl_setLocationTail != "")
        return rjs_XUrl_setLocationTail;
    else
        return str;
}

/*****************************************************
 * Begin inserting "rmi_setCookie(*)"
 *****************************************************/
function rjs_xCookieBegin(str)
{
    rjs_XCookie_on = true;

    // remove "o.document.cookie" in  "o.document.cookie ="
    var cur = rjs_Tokens.length-1;
    rjs_Tokens.null_section(rjs_Index_id, cur);

    return str;
}

/*****************************************************
 * Finish inserting "rmi_setCookie(*)"
 *****************************************************/
function rjs_xCookieEnd(str)
{
    rjs_XCookie_on = false;
    return str;
}

/*****************************************************
 * Begin inserting "rmi_xlateURL(*)" for "*.action="
 *****************************************************/
function rjs_xActionBegin(str)
{
    rjs_XAction_on = true;
    return str;
}
```

```
/*****************************************************
 * Finish inserting "rmi_xlateURL(*)" for "*.action="
 *****************************************************/
function rjs_xActionEnd(str)
{
    rjs_XAction_on = false;
    return str;
}

/*****************************************************
 * Begin inserting "rmi_xlate(*)"
 *****************************************************/
function rjs_xInnerHtmlBegin(str)
{
    rjs_XInnerHtml_on = true;
    return str;
}

/*****************************************************
 * Finish inserting "rmi_xlate(*)"
 *****************************************************/
function rjs_xInnerHtmlEnd(str)
{
    rjs_XInnerHtml_on = false;
    return str;
}


/*****************************************************
 * Translate "document.layers"
 *****************************************************/
function rjs_xLayers(str)
{
    if (rjs_LayerState != "doc") return str;

    var pre = rjs_Tokens.length-3;
    var cur = rjs_Tokens.length-1;
    if (pre < 0 || cur < 0) return str;

    if (rjs_Tokens[pre] == "document" && rjs_Tokens[cur] == "layers")
    {
        rjs_Tokens[pre] = "document.layers[\'rmilayer\'].document";
        rjs_LayerState = "";
    }

    return str;
}


/*************************************************************
 * Save current token position into a global variable
 * (e.g. rjs_Index_id)
 *************************************************************/
var rjs_Index_id = 0;

function rjs_saveIndexFor(type)
```

```
{
    var code = "rjs_Index_" + type + " = rjs_Tokens.length - 1";
    eval(code);
}

/*************************************************
 * Increment top elements for ALL 'nesting' arrays
 *************************************************/
function rjs_incTopForNesting()
{
    rjs_incTop(rjs_XUrl_nesting);
    rjs_incTop(rjs_XCookie_nesting);
    rjs_incTop(rjs_XAction_nesting);
    rjs_incTop(rjs_XInnerHtml_nesting);
}

/*************************************************
 * Decrement top elements for ALL 'nesting' arrays
 *************************************************/
function rjs_decTopForNesting()
{
    rjs_decTop(rjs_XUrl_nesting);
    rjs_decTop(rjs_XCookie_nesting);
    rjs_decTop(rjs_XAction_nesting);
    rjs_decTop(rjs_XInnerHtml_nesting);
}

/*************************************************
 * Increment the top element of an array
 *************************************************/
function rjs_incTop(array)
{
    var cur = array.length - 1;
    if (cur < 0) cur = 0;

    return ++array[cur];
}

/*************************************************
 * Decrement the top element of an array
 *************************************************/
function rjs_decTop(array)
{
    var cur = array.length - 1;
    if (cur < 0)  cur = 0;
    return --array[cur];
}

/*************************************************
 * Return (NOT pop!) the top element of an array
 *************************************************/
function rjs_retTop(array)
{
    var cur = array.length - 1;
    if (cur < 0)  cur = 0;
    return array[cur];
}
```

```
/*************************************************
 * Save head & tail positions for a chain
 * (e.g. for "a.b.c.d", positions of a & d are saved)
 *
 * Previous 'identifier' position is saved as a head position
 * Relative offset from current position is saved as a tail position
 *************************************************/
function rjs_saveChain(rel, head, tail)
{
    var cur = rjs_Tokens.length - 1;
    var pre = cur + rel;

    if (pre < 0 || cur < 0) return false;

    head.push(rjs_Index_id);
    tail.push(pre);


    return true;
}

/*********************************************
 * Save *.open() attributes
 *
 * Trigger State: *.open(
 *********************************************/
function rjs_saveOpen()
{
    var cur = rjs_Tokens.length - 1;
    var pre = cur - 1;
    if (pre < 0 || cur < 0) return false;

    if (rjs_Tokens[pre] == "open")
    {
        // e.g. save positions for a & c for "a.b.c.open("
        rjs_saveChain( -3, rjs_Open_head, rjs_Open_tail);

        rjs_OpenFunc_pos.push(pre);       // e.g. save position for
"open"

    }

    return true;
}

/*********************************************
 * Translate open(*) or *.open(*)
 *
 * Trigger State: * . open (*
 *********************************************/
function rjs_xOpen()
{

    if (rjs_OpenFunc_pos.length == 0) return false;

    var func_pos = rjs_OpenFunc_pos.pop();
```

```
    if (rjs_Tokens[func_pos-1] == ".")
    {
        var head_pos = rjs_Open_head.pop();
        var tail_pos = rjs_Open_tail.pop();

        rjs_Tokens[func_pos] = "rmi_winobj_open";
        var arg0 = rjs_Tokens.section(head_pos, tail_pos);
        rjs_Tokens[func_pos+2] = arg0 + ", " + rjs_Tokens[func_pos+2];
        rjs_Tokens.null_section(head_pos, tail_pos + 1);
    }
    else
        rjs_Tokens[func_pos] = "rmi_window_open";

}

/*******************************************
 * Save *.write() & *.writeln() attributes
 *
 * Trigger State: .write( or .writeln(
 *******************************************/
function rjs_saveWrite()
{
    var cur = rjs_Tokens.length - 1;
    var pre0 = cur - 1;
    var pre1 = cur - 2;
    if (pre0 < 0 || pre1 < 0 || cur < 0) return false;

    if (rjs_Tokens[pre1] == ".")
        if (rjs_Tokens[pre0] == "write" || rjs_Tokens[pre0] ==
"writeln")
        {
            // e.g. save positions for a & c for "a.b.c.write("
            rjs_saveChain( -3, rjs_Write_head, rjs_Write_tail);

            rjs_WriteFunc_pos.push(pre0);       // e.g. save position
for "write" or "writeln"
        }

    return true;
}

/*******************************************
 * Translate  *.write(*) or *.writeln(*)
 *
 * Trigger State: .write( or .writeln(
 *******************************************/
function rjs_xWrite()
{
    if (rjs_WriteFunc_pos.length == 0) return false;

    var func_pos = rjs_WriteFunc_pos.pop();

    if (rjs_Tokens[func_pos-1] == ".")
    {
        var head_pos = rjs_Write_head.pop();
        var tail_pos = rjs_Write_tail.pop();
```

```
        rjs_Tokens[func_pos] = "rmi_" + rjs_Tokens[func_pos];    // xlate
write or writeln
        var arg0 = rjs_Tokens.section(head_pos, tail_pos);
        rjs_Tokens[func_pos+2] = arg0 + ", " + rjs_Tokens[func_pos+2];
        rjs_Tokens.null_section(head_pos, tail_pos + 1);
    }
}

/*******************************************
 * Save *.location.replace() attributes
 *
 * Trigger State: *.replace(
 *******************************************/
function rjs_saveReplace()
{
    var cur = rjs_Tokens.length - 1;
    var pre0 = cur - 1;
    var pre1 = cur - 3;
    if (pre0 < 0 || pre1 < 0 || cur < 0) return false;

    if (rjs_Tokens[pre1] == "location" && rjs_Tokens[pre0] == "replace")
    {
        // e.g. save positions for a & c for "a.b.c.location.replace("
        rjs_saveChain(-5, rjs_Replace_head, rjs_Replace_tail);

        rjs_ReplaceFunc_pos.push(pre0);      // e.g. save position for
"replace"
    }

    return true;
}

/*******************************************
 * Translate location.replace(*) or *.location.replace(*)
 *
 * Trigger State: * . replace (*
 *******************************************/
function rjs_xReplace()
{
    if (rjs_ReplaceFunc_pos.length == 0) return false;

    var func_pos = rjs_ReplaceFunc_pos.pop();
    var head_pos = rjs_Replace_head.pop();
    var tail_pos = rjs_Replace_tail.pop();

    if (rjs_Tokens[func_pos-3] == ".")
    {
        // Handle the argument IF top or parent is in the chain

        var top_pos = rjs_findLast(head_pos, tail_pos, "top");
        var parent_pos = rjs_findLast(head_pos, tail_pos, "parent");

        var arg0 = "";

        if (top_pos != -1 || parent_pos != -1)
        {
```

```
            var split_pos = head_pos;

            if (top_pos == -1)
                split_pos = parent_pos;
            else if (parent_pos == -1)
                split_pos = top_pos;
            else
                split_pos = top_pos;   // use 'top' if both are found

            var win = rjs_Tokens.section(head_pos, split_pos);
            var rest = rjs_Tokens.section(split_pos+1, tail_pos);
            arg0 = "rmi_getTop(" + win + ")" + rest;
        }
        else
            arg0 = rjs_Tokens.section(head_pos, tail_pos);

        rjs_Tokens[func_pos] = "rmi_replace";
        rjs_Tokens[func_pos+2] = arg0 + ", " + rjs_Tokens[func_pos+2];
        rjs_Tokens.null_section(head_pos, tail_pos + 3);    // remove
chain.location.
    }
    else
    {
        rjs_Tokens[func_pos+2] = "self, " + rjs_Tokens[func_pos+2];
        rjs_Tokens[func_pos] = "rmi_replace";
        rjs_Tokens.null_section(head_pos, tail_pos + 3);    // remove
chain.location.
    }
}

/*******************************************
 * Save attributes for document.domain or *.document.domain
 *
 * Trigger State: *document.domain
 *******************************************/
function rjs_saveDomain()
{
    if (rjs_endsExactlyWith(rjs_Tokens.str() , "document.domain") )
    {
        // e.g. save positions for a & domain for
"a.b.c.document.domain("
        rjs_saveChain(0, rjs_Domain_head, rjs_Domain_tail);
    }

    return true;
}

/*******************************************
 * Pop attributes for document.domain or *.document.domain
 * for each assignment expression
 *
 * Trigger state: *document.domain in LHS
 *******************************************/
function rjs_popDomain()
{
    rjs_Domain_head.pop();
    rjs_Domain_tail.pop();
```

```
}

/*****************************************
 * Translate document.domain or *.document.domain
 *
 * Trigger State: end of statement
 *****************************************/
function rjs_xDomain()
{
    for (var i=0; i<rjs_Domain_head.length; ++i)
    {
        rjs_Tokens.null_section(rjs_Domain_head[i], rjs_Domain_tail[i]);
        rjs_Tokens[rjs_Domain_head[i]] = "rmi_getOriginalDomain()";
    }
}

/*****************************************
 * Save attributes for location.* or *.location.*
 *
 * Trigger State: *location.* or *location
 *****************************************/
function rjs_saveLocation()
{
    var cur = rjs_Tokens.length - 1;
    var pre0 = cur - 1;
    var pre1 = cur - 2;
    if (pre0 < 0 || pre1 < 0 || cur < 0) return false;

    var str = rjs_Tokens.str();
    var peek = JSC$parser_peek_token_token;

    // if (rjs_Tokens[pre1] == "location" && rjs_Tokens[pre0] == ".")

    if (rjs_endsExactlyWith(str, "location.href")
        || rjs_endsExactlyWith(str, "location.host")
        || rjs_endsExactlyWith(str, "location.hostname")
        || rjs_endsExactlyWith(str, "location.pathname")
        || rjs_endsExactlyWith(str, "location.port")
        || rjs_endsExactlyWith(str, "location.search")
       )
    {
        // e.g. save positions for a & href for "a.b.c.location.href"
        rjs_saveChain(0, rjs_Location_head, rjs_Location_tail);
    }
    else if (rjs_Tokens[cur] == "location" && peek != ".")
    {
        // e.g. save positions for a & location for "a.b.c.location"
        rjs_saveChain(0, rjs_Location_head, rjs_Location_tail);
    }

    return true;
}

/*****************************************
 * Save attributes for standalone 'location'
 *
 * Trigger State: location
```

```
 * (no '.location' or 'location.')
 ******************************************/
function rjs_saveStandaloneLocation()
{
    var cur = rjs_Tokens.length - 1;
    var pre = cur - 1;
    if (pre < 0 || cur < 0) return false;

    if (rjs_Tokens[rjs_Index_id] == "location" && rjs_Index_id == cur &&
rjs_Tokens[pre] != ".")
        rjs_saveChain(0, rjs_Location_head, rjs_Location_tail);
}

/*******************************************
 * Pop attributes for location.* or *.location.*
 * for each assignment expression
 *
 * Trigger state: *location.* in LHS
 *******************************************/
function rjs_popLocation()
{
    rjs_Location_head.pop();
    rjs_Location_tail.pop();
}

/*******************************************
 * Translate *.location, location.*, *.location.*
 *
 * Trigger State: end of statement
 *******************************************/
function rjs_xLocation()
{

    for (var i=0; i<rjs_Location_head.length; ++i)
    {
        var head_pos = rjs_Location_head[i];
        var tail_pos = rjs_Location_tail[i] - 2;
        var arg0 = rjs_Tokens.section(head_pos, tail_pos);

        var prop = rjs_Tokens[rjs_Location_tail[i]];

        if (prop == "location" && arg0 != "")    // from *.location
        {
            arg0 = arg0 + ".location";
            prop = "";
        }
        else if (prop == "location")             // from location
        {
            arg0 = "location";
            prop = "";
        }

        rjs_Tokens.null_section(rjs_Location_head[i],
rjs_Location_tail[i]);
        rjs_Tokens[rjs_Location_head[i]] = "rmi_getOriginal(" + arg0 +
", \"" + prop + "\")";
    }
```

```
    rjs_Location_head.reset();
    rjs_Location_tail.reset();
}

/*******************************************
 * Save attributes for *document.cookie*
 *
 * Trigger State: *document.cookie*
 *******************************************/
function rjs_saveCookie()
{
    var str = rjs_Tokens.str();

    if (rjs_endsExactlyWith(str, "document.cookie"))
    {
        // e.g. save positions for a & cookie for
"a.b.c.document.cookie"
        rjs_saveChain(0, rjs_Cookie_head, rjs_Cookie_tail);
    }

    return true;
}

/*******************************************
 * Pop attributes for *document.cookie*
 * for each assignment expression
 *
 * Trigger state: *document.cookie* in LHS
 *******************************************/
function rjs_popCookie()
{
    rjs_Cookie_head.pop();
    rjs_Cookie_tail.pop();
}

/*******************************************
 * Translate *document.cookie*
 *
 * Trigger State: end of statement
 *******************************************/
function rjs_xCookie()
{
    for (var i=0; i<rjs_Cookie_head.length; ++i)
    {
        var head_pos = rjs_Cookie_head[i];
        var tail_pos = rjs_Cookie_tail[i];
        var arg0 = rjs_Tokens.section(head_pos, tail_pos);

        rjs_Tokens.null_section(rjs_Cookie_head[i], rjs_Cookie_tail[i]);
        rjs_Tokens[rjs_Cookie_head[i]] = "rmi_getCookie(" + arg0 + ")";
    }

    rjs_Cookie_head.reset();
    rjs_Cookie_tail.reset();
}
```

```
/*******************************************
 * Save attributes for *.frames[*].*
 *
 * Trigger State: *.frames[
 *******************************************/
function rjs_saveFrames()
{
    var cur = rjs_Tokens.length - 1;
    var pre = cur - 1;
    if (pre < 0 || pre-1 < 0 || cur < 0) return false;

    if (rjs_Tokens[pre] == "frames" && rjs_Tokens[pre-1] == ".")
    {
        // e.g. save positions for a & c for "a.b.c.frames["
        rjs_saveChain( -3, rjs_Frames_head, rjs_Frames_tail);

        rjs_FramesObj_pos.push(pre);       // e.g. save position for
"frames"
    }

    return true;
}

/*******************************************
 * Translate *.frames[*].*
 *
 * Trigger State: *.frames[
 *******************************************/
function rjs_xFrames()
{
    if (rjs_FramesObj_pos.length == 0) return false;

    var obj_pos = rjs_FramesObj_pos.pop();    // "frames" position

    if (rjs_Tokens[obj_pos-1] == ".")
    {
        var head_pos = rjs_Frames_head.pop();
        var tail_pos = rjs_Frames_tail.pop();

        var left_pos = obj_pos+1;                          // left bracket
position
        var right_pos = rjs_findNext(left_pos, "]");

        if (right_pos != -1)
        {
            rjs_Tokens[obj_pos] = "rmi_getFrame";
            var arg0 = rjs_Tokens.section(head_pos, tail_pos);

            rjs_Tokens[left_pos] = "(" + arg0;

            if (right_pos - left_pos > 1)
                rjs_Tokens[left_pos] += ", ";
            else
                rjs_Tokens[left_pos] += ", 0";    // frames[]

            rjs_Tokens[right_pos] = ")";
```

```
                rjs_Tokens.null_section(head_pos, tail_pos + 1);
            }
        }
    }

    /*******************************************
     * Translate JavaScript string
     *******************************************/
    function rmi_xjs(str)
    {
        rjs_Error = false;          // reset

        JSC$generate_debug_info = false;
        JSC$warn_missing_semicolon = false;
        JSC$verbose = false;
        JSC$optimize_constant_folding = false

        var sStr = new JSC$StreamString(str);

        rjs_Stmts.reset();
        JSC$parser_reset ();

        JSC$parser_parse(sStr);

        rjs_debug("OLD:" + str + "\n" + "NEW:" + rjs_Stmts.str() );

        if (rjs_Error)
            return str;
        else
            return rjs_Stmts.str();
    }
```

APPENDIX B3

```
/*************************************************************
 *
 * U.S. Patent Pending. Copyright 2000 Yahoo! Inc.,
 * 3420 Central Expressway, Santa Clara, California U.S.A.
 *
 * ALL RIGHTS RESERVED.
 *
 * This computer program is protected by copyright law and
 * international treaties. Unauthorized reproduction or
 * distribution of this program, or any portion of it, may
 * result in severe civil and criminal penalties, and will
 * be prosecuted to the maximum extent possible under the law.
 *
 *************************************************************/


/*************************************************************
 * Add the following functions for built-in Array class
 *************************************************************/

    // Push an element into a stack
Array.prototype.push = function (obj)
{
    this[this.length++] = obj;
}

    // Pop an element into a stack
Array.prototype.pop = function ()
{
    var ret = this[this.length-1];
    if (this.length > 0) --this.length;

    return ret;
}

    // Reset a stack
Array.prototype.reset = function ()
{
    this.length = 0;
}

    // Return a string after joining all elements
Array.prototype.str = function ()
{
    var str = "";
    for (var i=0; i<this.length; ++i)
    {
        str += this[i];
    }

    return str;
}

    // Extract a section of array (including 'from' & 'to' elements)
    // Return a string after joining extracted elements
```

```
    // If "to" not specified, extract until the last element.
Array.prototype.section = function (from, to)
{
    if (typeof to == "undefined") to = this.length;

    var str = "";
    for (var i=from; i<=to && i<this.length; ++i)
    {
        str += this[i];
    }

    return str;
}

    // Null a section of array (including 'from' & 'to' elements)
    // If "to" not specified, nullify until the last element.
Array.prototype.null_section = function (from, to)
{
    if (typeof to == "undefined") to = this.length;

    for (var i=from; i<=to && i<this.length; ++i)
    {
        this[i] = "";
    }
}

/*************************************************************
 * Codes added to work with NGS
 *************************************************************/
function FileClass() {};

FileClass.prototype.byteToString = function (ch)
{
    return ch;
}

var File = new FileClass();

/*************************************************************
 * Codes added to work with NGS
 *************************************************************/
function error (msg)
{
    return rjs_error(msg);
}


/*************************************************************
 * Alert utilities
 *************************************************************/
function rjs_alert(str)
{
    alert(str);
}

function rjs_debug(str)
{
```

```
    // alert("debug: " + str);
}

function rjs_info(str)
{
    // alert("info: " + str);
}

function rjs_warn(str)
{
    // alert("warning: " + str);
}

function rjs_error(str)
{
    alert("RJS error: " + str);

    rjs_Error = true;
    return false;
}

/************************************************************
 * Global variables
 ************************************************************/

var rjs_Error = false;              // error status
var rjs_Tokens = new Array();       // tokens of a statement
var rjs_Stmts = new Array();        // all statements


var rjs_AssignmentState = "";       // "", lhs, rhs of =
var rjs_BracketState = "out";       // in, out of []
var rjs_CallState = "";
var rjs_LayerState = "";            // "", doc

var rjs_XUrl_on = false;            // for inserting "rmi_xlateURL(*)"
var rjs_XUrl_setLocationTail = "";  // stores a matching tail for
"rmi_setLocation("

var rjs_XCookie_on = false;         // for inserting "rmi_setCookie(*)"
var rjs_XAction_on = false;         // for inserting "rmi_xlateURL(*)"
for "*.action="
var rjs_XInnerHtml_on = false;      // for inserting "rmi_xlate(*)"

/************************************************************
 * Global variables - stacks storing token positions
 *
 * Each element of rjs_*_head stores the 1st token position of a chain
 * Each element of rjs_*_tail stores the last token position of a chain
 * Each element of rjs_*Func_pos stores the function position
 * Each element of rjs_*_nesting stores a counter for tracking nesting
'('
 ************************************************************/

var rjs_Open_head = new Array();
var rjs_Open_tail = new Array();
var rjs_OpenFunc_pos = new Array();
```

```
var rjs_Write_head = new Array();
var rjs_Write_tail = new Array();
var rjs_WriteFunc_pos = new Array();

var rjs_Frames_head = new Array();
var rjs_Frames_tail = new Array();
var rjs_FramesObj_pos = new Array();

var rjs_Replace_head = new Array();
var rjs_Replace_tail = new Array();
var rjs_ReplaceFunc_pos = new Array();

var rjs_Domain_head = new Array();
var rjs_Domain_tail = new Array();

var rjs_Location_head = new Array();
var rjs_Location_tail = new Array();

var rjs_Cookie_head = new Array();
var rjs_Cookie_tail = new Array();

var rjs_XUrl_nesting = new Array();
var rjs_XCookie_nesting = new Array();
var rjs_XAction_nesting = new Array();
var rjs_XInnerHtml_nesting = new Array();
```

APPENDIX C

<Function Calls>	
open(URL, TARGET, OPT)	rmi_window_open(URL, TARGET, OPT)
OBJ.open(URL, TARGET, OPT)	rmi_winobj_open(OBJ, URL, TARGET, OPT)
OBJ.write(S1, S2, ...)	rmi_write(OBJ, S1, S2, ...)
OBJ.writeln(S1, S2, ...)	rmi_writeln(OBJ, S1, S2, ...)
OBJ.location.replace (URL)	rmi_replace(OBJ, URL)
location.replace (URL)	rmi_replace(self, URL)
OBJ.top.MEMBER.location.replace(URL)	rmi_replace(rmi_getTop(OBJ.top).MEMBER, URL)
OBJ.parent.MEMBER.location.replace(URL)	rmi_replace(rmi_getTop(OBJ.parent).MEMBER, URL)

< Variables in left-hand-side of an assignment expression>	
OBJ.innerHTML = HTML	OBJ.innerHTML = rmi_xlate(HTML)
location = URL	location = rmi_xlateURL(URL)
location.href = URL	location.href =rmi_xlateURL(URL)
OBJ.location = URL	OBJ.location = rmi_xlateURL(URL)
OBJ.location.href = URL	OBJ.location.href =rmi_xlateURL(URL)
OBJ.action = URL	OBJ.action = rmi_xlateURL(URL)
OBJ.top.MEMBER.location.PROP = URL	rmi_setLocation("OBJ.top","MEMBER.location. PROP", rmi_xlateURL(URL),OBJ.top.MEMBER)
OBJ.parent.MEMBER.location.PROP = URL	rmi_setLocation("OBJ.parent","MEMBER.location. PROP", rmi_xlateURL(URL),OBJ.parent.MEMBER)

<Cookie setting>	
OBJ.document.cookie = STR	rmi_setCookie("OBJ.document.cookie", STR)

<Variables NOT in left-hand-side of an assignment expression>	
location	rmi_getOriginal(location, "")
location.href	rmi_getOriginal(location, "href")
location.host	rmi_getOriginal(location, "host")
location.hostname	rmi_getOriginal(location, "hostname")
location.pathname	rmi_getOriginal(location, "pathname")
location.port	rmi_getOriginal(location, "port")
location.search	rmi_getOriginal(location, "search")
OBJ.location	rmi_getOriginal(OBJ.location, "")
OBJ.location.href	rmi_getOriginal(OBJ.location, "href")
OBJ.location.host	rmi_getOriginal(OBJ.location, "host")
OBJ.location.hostname	rmi_getOriginal(OBJ.location, "hostname")
OBJ.location.pathname	rmi_getOriginal(OBJ.location, "pathname")
OBJ.location.port	rmi_getOriginal(OBJ.location, "port")
OBJ.location.search	rmi_getOriginal(OBJ.location, "search")
document.domain	rmi_getOriginalDomain()
OBJ.document.domain	rmi_getOriginalDomain()

<Cookie reading>	
OBJ.document.cookie	rmi_getCookie(OBJ.document.cookie)
OBJ.document.cookie.MEMBER	rmi_getCookie(OBJ.document.cookie).MEMBER

<Variables in any expression>	
OBJ.frames[INDEX].PROP	rmi_getFrame(OBJ, INDEX).PROP
document.layers	document.layers['rmilayer'].document.layers

SF 1125826 v1